Robust Optimisation of Computationally Expensive Computational Fluid Dynamics Models using Multi-fidelity Approaches

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Matthew Ellison

September 2021

Abstract

This dissertation explores the optimisation of computationally expensive models, whilst taking into account input uncertainty. The methods proposed are designed to be applicable to models used within the engineering industry, where a key aspect is the process of selecting a design that satisfies several constraints and performance objectives simultaneously. Modern engineering products often have to balance performance against factors such as profitability and environmental impact. There are often many feasible designs that satisfy these requirements. Locating such designs, and subsequently selecting an optimal choice, is often a challenging task. Moreover, any final design choice is subject to a variety of uncertainty that arises in both the manufacture and life cycle of the product. Selecting a design that is robust to such uncertainty, whilst still exhibiting near-optimal performance, is critical to the lifetime performance of a new product.

Complex computer models are increasingly employed in the design process to provide information on the estimated performance of a potential design. Such models are typically computationally expensive, which often limits the number of evaluations available to perform tasks such as robust optimisation. Two novel approaches are presented in this work to address this problem. The first is a direct optimisation approach extending an algorithm known as subset simulation to factor in input uncertainty, alongside strategies to boost its computational efficiency. In general, this is the preferred approach as it introduces no further uncertainty into the problem, however in the case that computational constraints prohibits its application, another approach is necessary. This provided the motivation behind the second approach, which employs a surrogate modelling technique known as Gaussian process emulation to provide an inexpensive statistical approximation of the expensive computer model. This emulator is enhanced with a novel sampling scheme and multi-fidelity training data, and is optimised in place of the expensive computer model without the computational constraints. As such, the approaches are not in direct competition, but provide the means to perform robust optimisation across a range of computational budgets. The theoretical underpinnings of each of the proposed methods are discussed in detail, before they are applied to illustrative examples. Finally, each of the methods are applied to industrial case studies involving expensive computational fluid dynamics models provided by the industrial partner. The results showed that the two approaches were successful in performing efficient robust optimisation of computational expensive engineering models. In particular, the direct approach results showcased the considerable impact on the computational efficiency of the robust optimisation process, without compromising on performance. For the surrogate approach, the case studies highlight the ability to successfully perform robust optimisation even with stringent computational constraints.

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed Matthew Ellison
Date 31/03/2021

STATEMENT: This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed Matthew Ellison
Date 31/03/2021

Acknowledgements

First and foremost, I would like to thank my supervisor Dr Alejandro Diaz De la O. The impact of having a supportive and enthusiastic supervisor cannot be overstated. I am extremely fortunate that Alex has had both of these qualities in abundance during our time together. I will always be grateful for his guidance and patience on my PhD journey.

I wish to gratefully acknowledge the financial support contributed towards the funding of my studies by the EPSRC and General Electric Power.

I would like to thank everyone at General Electric Power who I had the pleasure to work with over the course of the PhD. In particular; a massive thank you to Dr Andrew Pike for his support early on in the PhD, Dr Greg Laskowski for his help with the industrial case study, Dr Nadir Ince and Mark Willetts for their continued help, patience and engagement throughout my studies.

I would like to thank Dr Anas Batou and Dr Mauro Innocente for reading this dissertation and providing several useful comments and suggestions.

I would also like to express my gratitude to my colleagues at the Institute for Risk and Uncertainty, as well as the University of Liverpool at large. In particular, special thanks to Peter Hristov, Alfredo Garbuno Iñigo and Paul Byrnes whose advice and assistance were invaluable during my experience.

I would like to give my deepest thanks to my family and friends who have supported me throughout the years, and without whom I am certain I would not have got this far. To my brother Daniel, thank you for your continued assistance throughout the PhD, for patiently listening whenever I needed to vent and for picking me up whenever I was down. To my long-suffering girlfriend Lauren, thank you for bearing the brunt of a pandemic, a never-ending thesis and a grumpy companion for far too long. Your understanding and support got me through the most trying experience of my life and I am eternally grateful. Finally, to my mum and dad, I will forever be indebted to the sacrifices you have made for me throughout my life. You inspire me each and every day to be the best I can be.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

GE General Electric

LF Lower Fidelity

HF Higher Fidelity

CFD Computational Fluid Dynamics

RANS Reynolds-Averaged Navier Stokes

LES Large-Eddy Simulation

DNS Direct Numerical Simulation

UQ Uncertainty Quantification

SOP Single-Objective Optimisation Problem

MOP Multi-Objective Optimisation Problem

GA Genetic Algorithm

PSO Particle Swarm Optimisation

SuS Subset Simulation

i.i.d independent and identically distributed

MC Monte Carlo

MCMC Markov Chain Monte Carlo

MMH	Modified Metropolis Hastings
PDF	Probability Density Function
CDF	Cumulative Density Function
MF	Multi-Fidelity
ACO	Ant Colony Optimisation
RMOP	Robust Multi-Objective Optimisation Problem
RSS	Robust Subset Simulation
MF-RSS	Multi-Fidelity Robust Subset Simulation
CST	Class Shape Transformation
L/D	Lift-to-drag
GPE	Gaussian Process Emulator
DOE	Design of Experiments
LHS	Latin Hypercube Sampling
RSM	Response Surface Methods
NN	Neural Networks
SVM	Support Vector Machines
ARD	Automatic Relevance Determination
LOO-CV	Leave-one-out Cross Validation
MLE	Maximum Likelihood Estimation
RGPE	Robust Gaussian Process Emulator
SF	Single-Fidelity
HVEI	Hypervolume Expected Improvement

MF RGPE Multi-Fidelity Robust Gaussian Process Emulator

Chapter 1

Introduction

1.1 Motivation

Engineers are synonymous with formulating creative solutions to complicated problems. Some of the greatest feats in the history of humanity can be attributed to their rationale and innovation. The goal of engineering design in particular is to create technological designs that satisfy specific performance objectives and constraints over a period of time. Modern engineering designs often have to balance performance of a potential design against factors such as profitability and environmental impact. There are often many feasible choices that satisfy these requirements. Locating such designs, and subsequently selecting an optimal choice, is a challenging and crucial task. Traditionally, the selection of a design has involved attempting to locate a design with the highest nominal performance according to some criterion, such as power output, efficiency, amongst others. However, simply optimising for nominal performance can oversimplify the problem and actually result in a suboptimal design. Keane and Nair [1] point out that optimising for nominal performance often fails to take into account the uncertainties that arise in modelling, manufacturing and operation. Moreover, once in operation, such designs usually suffer a sharper decline in performance due to degradation compared to other suboptimal designs. Figure 1.1 illustrates this concept for various components of a turbine blade.

The objective of robust design is to determine a set of designs that exhibit high levels of

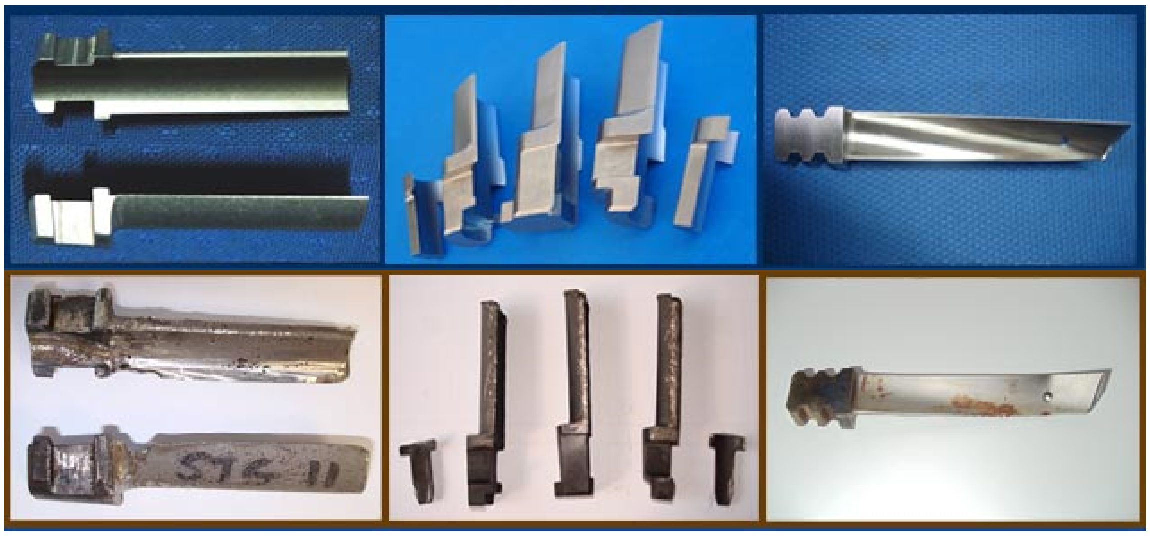
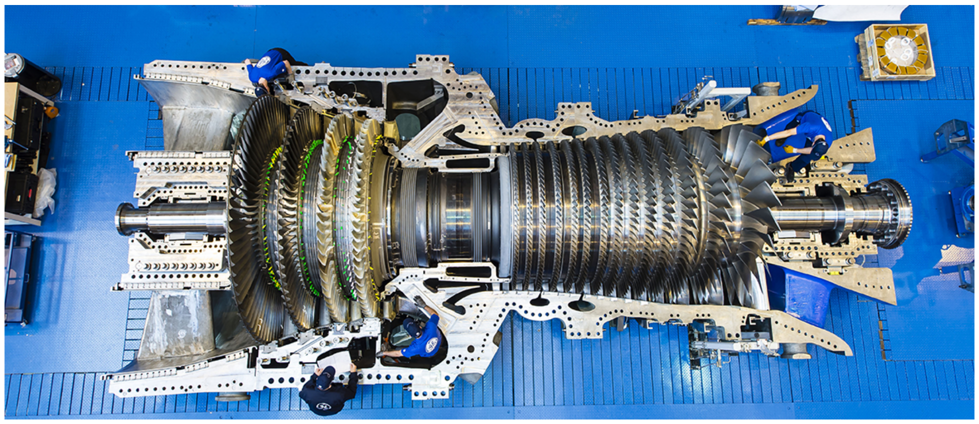


Figure 1.1: **Degradation of Turbine Blades:** A comparison of several turbine blades from the original design concept to the end of their life cycle. Taken from [2].

performance with low variability, whilst taking uncertainties into account. The benefits of robust design include the assurance of high performance regardless of a variety of unknown factors and occurrences throughout the life cycle. The concept of robust design is often attributed to Taguchi, who argued that designs should be as insensitive as possible to variations outside of the designer's control, and developed a method based on loss functions to facilitate this into the design procedure [3, 4]. The Taguchi method has received significant criticism and is generally considered to be outdated for modern day applications [5]. However, the philosophy behind the method, i.e. factoring in uncertainty and noise into the design process, has become increasingly popular and crucial in recent years.

A perfect example is General Electric (GE), who have pioneered technologies and provided world-leading products in areas ranging everywhere from transportation to healthcare for over 125 years. In particular, GE products such as those displayed in figure 1.2, currently generate a third of the world's electricity [6]. However, GE operate in an essentially Darwinian environment of survival of the fittest. Consequently, to maintain their position as a global leader in areas such as power generation, they have to continue to manufacture products that consistently exhibit high performance throughout lengthy life cycles. Given the intricate components and extreme operating conditions of GE's products, factoring these uncertainties into the design process is critical. A key characteristic of this design process, and indeed practically all modern engineering design processes, is the use of computational models, to replace or aid costly physical experiments.



(a)

(b)

Figure 1.2: Top: A General Electric H-Class gas turbine [7]. Bottom: Several GE Haliade 150-6MW offshore wind turbines [8].

1.2 Background

1.2.1 Computational modelling

Computational models, or simulators, are widespread throughout most areas of science and engineering. Such models typically employ mathematical descriptions to describe the key characteristics of the system, before utilising the processing power of computers to simulate how the system behaves. They offer a myriad of advantages over real-world experimentation, which can often be expensive, dangerous, unethical, or even impossible. Computational models essentially take a set of input variables $\mathbf{x}$, which may represent factors such as design geometry, to produce a set of outputs, or performance variables $\mathbf{y}$. Consequently, this mapping of input variables to performance variables can be considered as a mathematical function $\mathbf{y} = f(\mathbf{x})$. By adjusting the input variables, and monitoring the effect of the outputs, computational models can be used for a variety of purposes, such as optimisation of an engineering design.

In realistic industrial settings, the systems being modelled are often extremely complex. This complexity carries over to the associated computational models, such that the nature of f is not usually explicitly known. This means that any output for a specific set of input values is not known until they are evaluated on the model. Such computational models are generally referred to as 'black-box' models. Computational models can also be categorised as either stochastic or deterministic. Stochastic models produce different output values each time the model is run at the same set of inputs, due to some randomness caused by a stochastic component. Deterministic models on the other hand produce the same output values each time the same input is evaluated. Additionally, it is not uncommon for more than one computational model to exist for a given problem. There are often various mathematical descriptions available in the initial construction of the model, differing levels of abstraction, steady versus transient approximation, differing resolution, amongst many others. Each computational model possesses different assumptions, but ultimately they are all attempting to represent the same system. Consequently, in the scenario where multiple computational models are available, models are categorised into levels of fidelity: lower-fidelity (LF) models are usually defined by a lower-computational cost, but lower accuracy, than higher-fidelity (HF) models.

Regardless of the exact nature, a recurrent feature across a variety of computational models in engineering is a high computational expense involved in running the model. Such models are commonly referred to as computationally expensive models, and arise due to the fact that industrially-relevant models need to portray the behaviour of complex systems with extremely high accuracy. For instance, computationally expensive models have been utilised to aid in the wing design of aircraft [9, 10], improve the efficiency and robustness of turbine blades [11, 12], test the crash worthiness of vehicles [13] and estimate the reliability of crane design [14]. Although the definition of what classifies as computationally expensive can vary across different applications, a uniting feature is that such models can only be evaluated a limited number of times in industrial settings. This is as there are often strict time and cost limits involved in these settings before important and costly decisions must be made, e.g. a set time frame before the selection of a prototype design. Moreover, a potential fix for the problem is often nullified, as any increase in computing power and speed often leads to more complex models being developed to replace the previous versions. As such, the models remain computationally expensive even with the improved computing resources. The research presented in this thesis holds particular interest in computational fluid dynamics models, which exhibit many of the features mentioned above, and are discussed in more detail in the next subsection.

1.2.2 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) is the analysis of systems involving fluid flow, heat transfer and associated phenomena via computer-based simulation [15]. CFD involves discretizing the equations governing the behaviour of fluids into a system of algebraic equations, which are then solved to obtain an approximate solution that describes the behaviour of the aforementioned phenomena. CFD is widespread throughout engineering, and has been applied to problems such as analysing the aerodynamics of aircraft [16], hydrodynamics of ships [17] and combustion in gas turbines [18] to name a few. The typical steps involved in a CFD model are pre-processing, solving and post-processing. These steps are discussed in detail in [15], and summarised here.

The pre-processing stage begins by constructing the geometry involved in the problem, also known as the computational domain. This domain is then subdivided into a non-overlapping grid (or mesh) of cells, using a spatial discretization method. The shape of these cells is user-defined, with typical choices including hexahedral, tetrahedral, prismatic, pyramidal or polyhedral. Additionally, other key user decisions involve whether cell distribution within the mesh is uniform or non-uniform, i.e. evenly distributed or higher concentration in certain areas, and structured or unstructured. These choices directly affect computational efficiency and solution accuracy, and are often problem specific. Regardless of the exact configuration, a key influence on the accuracy of a CFD solution is the number of cells in a mesh. Generally, increasing the number of cells results in superior accuracy, but requires greater computational expense. The last step in this stage is defining the physical properties of the problem, such as particular fluid properties, and specifying any relevant initial conditions and boundary conditions.

The solution stage involves integrating the governing equations of fluid flow over each of the cells defined in the pre-processing stage. This is done by converting these equations into a set of algebraic equations through discretization, that can then be solved numerically using an iterative method. There are several choices to perform discretization, including finite difference [19], finite element [20] and spectral methods [21]. A popular choice amongst established CFD practices is to employ a particular form of finite difference known as the finite volume method [22]. This method begins by applying the integral form of the conservation equations to each cell, or control volume. The idea is that within a finite control volume a general flow variable, e.g. a velocity component, can be expressed as a balance between the various processes tending to increase or decrease it. Essentially, the rate of change of the general flow variable within the control volume is related to the net rate of increase/decrease due to convection, diffusion and creation/destruction of the variable. A computational node is located at the centroid of each control volume, at which the values of a general flow variable are calculated. The variable values are the surface of a control volume are then calculated using interpolation of the nodal values. Surface and volume integrals are approximated using suitable quadrature formulas. As a result, one obtains an algebraic equation for each control volume, in which a number of neighbour nodal values appear [23]. Finally, a suitable iterative method is used to solve. The methods employed in this stage must be able to deal with both laminar and turbulent flows. Laminar flows are characterised by fluid particles moving in smooth paths, with little to no mixing. Turbulent flows are characterised by chaotic behaviour of the flow such that large-scale eddies form due to flow instability. These large-scale eddies take energy from the mean flow and feed it down a cascade of progressively smaller eddies until it dissipates. Consequently, turbulent flows are much more complicated to predict and model than laminar flows. There are a variety of different methods to deal with this problem, however techniques can usually be summarised into the general categories of Reynolds-Averaged Navier Stokes equations (RANS), which models the entirety of the turbulent energy cascade, Large-Eddy Simulation (LES), which resolves some of the larger eddies but models the smaller eddies, or Direct Numerical Simulation (DNS), which resolves all of the eddies. The final stage is post-processing, in which the results of the solution stage are analysed depending on the particular nature of the problem.

1.2.3 Uncertainty in computational models

Computational models are exceptionally useful tools to investigate their given physical systems and phenomena. However, given the complex nature of such systems, it is often impossible for a model to capture the behaviour of the system completely. For example, models often possess parameters which require tuning to best approximate certain physical properties of the system [24]. This tuning task is often complicated, and there is usually no guarantee that a suitable value is selected. Consequently, this tuning task introduces a measure of uncertainty within the inputs of the model, which then propagates through to the model outputs. This is just one example of many potential sources of uncertainty that can be associated with computational models. The field of uncertainty quantification (UQ) has arisen in an attempt to identify and address these various uncertainties. The authors of [25] provide definitions of some of the common sources of uncertainty in computational models:

- Parameter uncertainty: already touched upon above, parameter uncertainty refers to the uncertainty surrounding the selection of suitable input values of the computational model. Inputs that represent known processes can be fixed, e.g. gravity can be set to $9.8\mathrm{m\,s}^{-2}$ with complete certainty, whereas the nature of some inputs will be unknown and may take a range of plausible values. Such inputs introduce a measure of uncertainty and may be represented with an upper and lower bound, a mean and a variance, or a probability distribution.

- Model discrepancy: The difference between the output of the computational model and the output of the actual physical system. This can be caused by factors such as approximations within the mathematical model, insufficient spatial or temporal resolution or simply errors in the computer code.

- Residual variability: consists of uncertainties that arise that cause the model output to vary between multiple evaluations of the same set of inputs. A potential cause is that the underlying physical process is actually stochastic, and the residual variation is a natural byproduct. Another cause is an insufficient number of inputs to the model allowing certain settings to vary between evaluations. Consequently, increasing the number of input variables may help to reduce the variability between evaluations.

- Parametric variability: The case whereby some input variables are left uncontrolled and unspecified (often deliberately), causing uncertainty that propagates through to the model output.

- Observational error: measurements of the actual physical system are used to calibrate or validate the computational model. These values often contain uncertainty bounds to account for any potential measurement error, which adds an extra source of uncertainty known as observational error.

- Code uncertainty: In essence, the model output for a given set of inputs is known, as the model is simply a known function of the inputs. However, due to the complex nature of the model, and the associated computational strain involved in evaluating it, the model output is not known until the model is actually evaluated. As it is often infeasible to evaluate all potential input values, uncertainty in the model output for untried input values must be accounted for and is known as code uncertainty.

A more general categorisation of the sources of uncertainty is provided in [26], whereby the type of uncertainty is dependent on whether it is inherent to the system under study, or simply arises due to lack of knowledge. The former is known as aleatoric, or irreducible, uncertainty, and represents uncertainty that is caused by natural variability within a system and cannot be reduced by any further information regarding the system, for example the outcome of flipping a coin. The other category is epistemic, or reducible, uncertainty, which arises due to some form of lack of knowledge, such as an overly simplistic model neglecting part of the underlying physical process. Such uncertainties are considered reducible as they can theoretically be reduced, and in some cases eliminated, given adequate additional information. Finally, it is possible for uncertainty to arise that can be a combination from both aleatoric and epistemic sources. For example, rolling a biased dice will always possess a random outcome (aleatoric uncertainty) but increasing the sample size would help reduce the uncertainty surrounding the associated probability of each outcome (epistemic uncertainty). Regardless of the definition, uncertainty is often abundant in a variety of engineering problems. As such, it is paramount to possess the tools to quantify any relevant uncertainty, and factor it into the solution of the problem.

1.2.4 Robust Optimisation

A common and critical problem within engineering is the task of optimisation. Optimisation is the process of locating the input configuration for a design that corresponds to optimal performance, which itself is usually defined by maximising or minimising some performance variable. Problems with a single performance variable under consideration are referred to as single-objective optimisation problems (SOPs). Problems with two or more objectives are described as multi-objective optimisation problems (MOPs). For MOPs, it is unusual for a single design to optimise each performance objective simultaneously, and as such most MOPs possess a set of optimal solutions known as the Pareto front [27]. A variety of methods have been developed to tackle MOPs, such as sequential quadratic programming [28], Quasi-Newton's methods [29], genetic algorithms (GA) [30], particle swarm optimisation (PSO) [31] and, used and covered in chapter 2 in this thesis, subset simulation [32].

The goal of traditional optimisation methods, such as those mentioned above, is to locate the global optima of the underlying objective function. However, as touched upon previously, such optima can often be sensitive to changes in their inputs or environment [33] and actually possess suboptimal performance. As discussed in section 1.2.3, there are a variety of uncertainties that can arise when employing computational models. Robust optimisation is the process of locating input variables that exhibit near-optimal performance and are insensitive, or robust, to the effect of the various uncertainties of the problem [34]. Traditional optimisation methods fail to address any uncertainties, and as such are unsuitable to perform robust optimisation. The research in this dissertation explores methods to tackle robust optimisation problems. A more thorough mathematical definition of the various optimisation problems is given in chapter 2 and chapter 3.

1.2.5 Surrogate Models

Optimisation tasks often involve significant evaluations of the underlying objective functions in order to converge towards regions with optimal performance. Given that these objective functions are often computationally expensive models, this presents a challenge. One solution that is explored in this thesis is to increase the efficiency of optimisation algorithms as a means to reduce the number of evaluations needed. Another approach is to employ a surrogate model, or metamodel, in place of the computationally expensive model. An incentive to use computational models rather than physical experiments was that the former is considerably less expensive, but still captures the behaviour of the physical process. Surrogate models provide a similar incentive when compared to computationally expensive models. A surrogate model attempts to approximate the behaviour of the more expensive model, but at a fraction of the computational cost. The surrogate is trained on a relatively small number of simulator evaluations, which provides information on the output space of the true model. This surrogate model can then provide an estimate of the true output at unknown input values, and thus be used in place of the computationally expensive model in the optimisation process, without the associated computational constraint. Given their merits, a plethora of different surrogate have been developed over the years. Amongst the most popular choices are neural networks [35], Taylor series expansion [36], radial basis functions [37] and support vector machines [38]. Another prevalent approach is Gaussian process emulation [39], which provides a distribution for the model output, supplying an approximation to the computational model as well as quantifying the output uncertainty associated with the use of the surrogate. For this and other reasons that will be outlined in Chapter 4, Gaussian process emulation and its extensions will be used for all surrogate modelling purposes in this research.

1.3 Objectives of this research

The preceding sections have highlighted some of the difficulties associated with the design process for modern engineering settings, as well as the potentially huge ramifications in the case of selecting a suboptimal design. Two of the main components contributing to these difficulties are the uncertainties that arise in both the manufacturing phase and life cycle of a design, and the computational expense related to the computer models employed in the design process. The research contained in this thesis aims to address these components, and develop methods to facilitate the efficient optimisation of computational expensive models whilst factoring in input uncertainty. To achieve this aim, the following objectives are identified and addressed in this dissertation:

1. The development of a novel robust optimisation algorithm that can be applied directly to computational models. Particular emphasis is given to ensure the algorithm is as computationally efficient as possible.
2. Provide a statistical surrogate modelling framework in the case that the computational model is too expensive for direct application of the robust optimisation algorithm. This framework can then be optimised in place of the expensive model.
3. Ensure that both approaches are suitable for industrial application and showcase the methods with relevant industrial case studies.

In particular, the subset simulation algorithm was identified as a suitable candidate to address the first objective. However, the method is currently suitable for nominal optimisation problems. This thesis intends to modify the method in order to adapt it for the purposes of robust optimisation. For the second objective, Gaussian process emulation was identified as an ideal surrogate modelling method for the computational model. This thesis intends to combine various enhancements of Gaussian process emulation to extend the applicability of the work to problems with severe computational constraints.

Upon meeting these objectives, a key criteria of the PhD is to ensure that the methods can be incorporated directly into the design process of the industrial partner. In particular, placing emphasis ensuring that the code is user-friendly, and easily understandable even for those without prior experience in MATLAB or in the concepts themselves.

1.4 Methodology

The key concepts of the subset simulation method were first covered, including the influence of its various hyperparameters. The distinction between nominal optimisation and robust optimisation was then presented in order to highlight the necessary modifications that would be required. To facilitate the use of subset simulation for robust optimisation, several modifications were proposed to address the two issues; factoring in input uncertainty and computational efficiency. The former was addressed by utilising neighbourhood samples to produce averaged objective values, including discussion on the impact of varying the number of neighbourhood samples. For the latter several strategies were proposed, including separate stages for nominal optimisation and robust optimisation, reducing unnecessary model evaluations by using a *bank* of solutions and employing adaptive Markov Chain Monte Carlo algorithm to boost efficiency. Further, the adapted subset simulation was also generalised to incorporate multiple levels of fidelity to further boost computational efficiency. In the absence of existing benchmark problems, an existing robust optimisation problem was extended to MF to facilitate testing of the method before industrial application.

Similarly for the surrogate approach, the key concepts of Gaussian process emulation were detailed, with particular attention to estimating hyperparameters, incorporating input uncertainty and boosting the quality of the training data through adaptive sampling and use of multi-fidelity data. Combining each of the enhancements, including a novel adaptive sampling scheme, for the purposes of enabling the use of the direct robust optimisation approach was then covered. A benchmark problem was then constructed and tackled.

Finally, both the direct approach and the surrogate approach were applied industrial case studies. Specifically, both were applied to optimising an aerofoil whilst taking into account input uncertainty, while the surrogate approach was also applied to a second case which involved optimising the design of a turbulated duct with uncertain input parameters. Unfortunately it was not possible to apply the direct approach to the turbulated duct case study due to technological restraints unrelated to the method itself.

1.5 Contributions

A novel method incorporating input uncertainty into the optimisation process was proposed, developed and tested on an industrially relevant problem. The algorithm, denoted robust subset simulation, extended the subset simulation algorithm to optimise over averaged objective values for a neighbourhood surrounding a particular set of input values. To ensure that the method was suitable for computationally expensive models, several measures were put in place to boost efficiency. Additionally, a second algorithm was also developed, denoted multi-fidelity robust subset simulation, which extended the robust subset simulation algorithm to utilise multiple levels of fidelity. The two related methods address the main issue facing existing direct approaches, providing a computationally responsible approach to perform robust optimisation of problems with one or more computationally expensive models.

In certain problems, computational constraints render all direct approaches infeasible, even with the aforementioned measures to boost efficiency. To address this, a surrogate modelling framework was constructed using various enhancements of Gaussian process emulation in order to facilitate the use of the robust subset simulation algorithm. This extended the applicability of the method to more computationally stringent problems. Moreover, in certain circumstances, the surrogate approach is able to incorporate uncertainty directly into the emulation process, allowing for use of a novel adaptive sampling scheme and enabling a nominal optimisation algorithm to be used, increasing the efficiency of the entire robust optimisation procedure.

All the methods discussed in this dissertation were incorporated in a MATLAB toolbox provided to the industrial partner. This included detailed documentation and template code to reproduce each of the industrial case studies. The toolbox is currently in the process of being transferred to the industrial partner's in-house design software for further application in real-world industrial problems.

1.6 Outline of Dissertation

The structure of the thesis is as follows:

Chapter 2 provides an in-depth description of subset simulation method. This begins with the original motivation and concept of the method, before discussing the reasoning and steps involved to adapt the method for the purposes of optimisation.

Chapter 3 presents a novel robust optimisation approach based on extending the subset simulation methods discussed in chapter 2. Particular attention is paid to the computational cost of the novel approach. The approach is then applied to an industrially relevant problem.

Chapter 4 introduces the concept of surrogate modelling, before highlighting Gaussian process emulation in particular. The theoretical and practical aspects of Gaussian processes are then covered, before detailing the steps involved to construct a Gaussian process emulator. Finally, several enhancements of Gaussian process emulation are discussed.

Chapter 5 presents a framework which combines the enhancements of Gaussian process emulation introduced in chapter 4 for the purposes of robust optimisation. The framework is then applied to two industrially relevant problems.

Finally, Chapter 6 concludes the thesis with a summary of the key findings from the preceding chapters, as well as giving directions for future development of the work.

Chapter 2

Subset Simulation

This chapter presents a thorough overview of subset simulation. The chapter begins by introducing the reliability problem that provided the original motivation behind the subset simulation method. The concept and theoretical background of the subset simulation algorithm is then discussed in detail. Next an analogy between reliability and optimisation is made to justify adapting the subset simulation algorithm for the purposes of optimisation. The steps involved to utilise the algorithm for single-objective optimisation are then covered, before a test problem is presented to illustrate the concept in action. Lastly, the steps to adapt the algorithm for multi-objective optimisation are discussed, including another illustrative test problem used to highlight the similarities and differences between the single-objective case. The optimisation algorithms presented within this chapter provide the foundation for the novel method discussed in chapter 3.

2.1 Background

Computational models are exceedingly common in engineering applications to model various natural processes, and are usually represented mathematically as $y = f(\mathbf{x})$. The model provides a mapping from a set of input variables to a number of output, or performance, variables, which can be utilised to learn more about the underlying characteristics of the system under study. A characteristic of particular importance in a number of fields is the reliability of a system. This reliability is typically associated with the probability of failure of a system, with a lower probability value indicating a more reliable system, and vice versa. Failure is typically defined as unacceptable performance of the system, according to some criteria. In a mathematical sense, failure is traditionally defined as the scenario where the performance of the system exceeds some predefined critical threshold. That is for a given critical threshold y^*, if $y = f(\mathbf{x}) < y^*$, the system is considered safe, or reliable, but if $y = f(\mathbf{x}) > y^*$, then the system fails and is considered unreliable. As a result, the failure domain of the system can be defined as the set of input values that correspond to a performance value that exceeds this critical threshold:

$$F = \{\mathbf{x} : f(\mathbf{x}) > y^*\}. \tag{2.1}$$

However, most engineering systems are complex, meaning that information regarding the system is often incomplete, resulting in uncertainty in the values of the input variables, which then propagates through to the performance variable. Consequently, input variables are modelled as random variables, whose marginal distributions are obtained from expert opinion, experimental data, or from literature. Given that $\pi(\mathbf{x})$ is the joint probability density function (PDF) for $\mathbf{x}$, the engineering reliability problem is then to compute the probability of failure, p_F:

$$p_F = P(\mathbf{x} \in F) = \int_F \pi(\mathbf{x}) d\mathbf{x}. \tag{2.2}$$

As the systems are often complicated, it is usually impossible to solve this integral analytically or numerically. There have been various methods developed over the years to address this issue, which can be roughly categorised into three groups. The first group contains analytical methods, such as the First-Order Reliability Method [40] and the Second-Order Reliability Method [41]. The second group involves any surrogate-based techniques, such as support machine vectors [42], neural networks [43] and response surface methods [44]. Finally the third group are approaches employing Monte Carlo (MC) simulation methods, including line sampling [45], importance sampling [46] and subset simulation [47]. Focusing on the third group specifically, the MC method [48] is an application of the law of large numbers, that computes the empirical mean as an approximation of the expected value of a quantity of interest. In particular, for some function $h : \Omega \to \mathbb{R}$, the expectation of the function $E_\pi[h(\mathbf{x})]$ with respect to the PDF $\pi(x)$ is

$$E_\pi[h(\mathbf{x})] = \int_\Omega h(\mathbf{x})\pi(\mathbf{x})d\mathbf{x}. \tag{2.3}$$

Given N independent and identically distributed (i.i.d) samples from $\pi(\mathbf{x})$, the MC method gives

$$E_\pi[h(\mathbf{x})] \approx \frac{1}{N}\sum_{i=1}^{N} h(\mathbf{x}^i). \tag{2.4}$$

To relate this to the reliability problem, the indicator function, $I_F(\mathbf{x})$, is defined as

$$I_F(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in F, \\ 0, & \mathbf{x} \notin F, \end{cases} \tag{2.5}$$

and p_F can be expressed as the expectation of the indicator function with respect to the joint probability distribution:

$$p_F = \int_F \pi(\mathbf{x})d\mathbf{x} = \int_\Omega I_F(\mathbf{x})\pi(\mathbf{x})d\mathbf{x} = \mathbb{E}_\pi[I_F(\mathbf{x})] \approx \frac{1}{N}\sum_{i=1}^{N} I_F(x^{(i)}) = p_F^{MC}. \tag{2.6}$$

Here p_F^{MC} denotes the MC estimation of failure, where $p_F^{MC} \rightarrow p_F$ as $N \rightarrow \infty$. On inspection, the MC approximation is simply the ratio of the total number of samples that result in failure to the total number of samples [49]. However, real-world reliability problems are synonymous with small failure domains, and consequently the MC approach is often too inefficient to be directly applied.

2.2 Subset Simulation for Reliability

Conceptually, subset simulation (SuS) addresses the issue of sampling from small-failure domains by modelling the failure region as a sequence of less-rare nested regions,

$$F = F_M \subset F_{M-1} \subset ... \subset F_1, \tag{2.7}$$

where sampling from F_1 is a relatively common event as described below. As a result, the probability of failure can then be defined as a product of conditional probabilities:

$$p_F = P(F_M) = P(F_1)P(F_2|F_1)...P(F_{M-1}|F_{M-2})P(F_M|F_{M-1}). \tag{2.8}$$

The SuS algorithm, as with many other reliability methods, assumes that any input variables $\mathbf{x}$ are independent. Moreover, $\mathbf{x}$ are assumed to be independent and identically distributed (i.i.d) Gaussian for convenience [49]. Consequently, transformation of $\mathbf{x}$ may be in order prior to the SuS algorithm commencing if they are not in this form. Depending on the initial nature of $\mathbf{x}$, this process may be done via methods such as the Nataf transformation [50] and the Rosenbatt transformation [51], or even standardization if $\mathbf{x}$ are already independent Gaussians. After this transformation is complete, the joint probability distribution follows the standard multivariate Gaussian distribution:

$$\pi(\mathbf{x}) = \pi(x_1, ..., x_d) = \prod_{i=1}^{d} \phi(x_i), \tag{2.9}$$

where $\phi(\cdot)$ represents the standard Gaussian probability distribution function. Once the input variables are i.i.d Gaussian, the SuS algorithm [47] begins by employing MC methods to generate n samples $\mathbf{x}_0^1, ..., \mathbf{x}_0^n \sim \pi(\mathbf{x})$ and computing their system performance values $y_0^1 = f(\mathbf{x}_0^1), ..., y_0^n = f(\mathbf{x}_0^n)$. Here the subscript 0 represents the fact that this is the zeroth level of the algorithm. The system performance values are then ordered from largest to smallest, renumbering the superscripts of the samples if necessary, such that $y_0^1 \geq ... \geq y_0^n$. Thus, the sample $\mathbf{x}_0^n$ can be considered the safest sample and $\mathbf{x}_0^1$ the sample closest to failure. Specifying $p \in (0, 1)$ such that np is an integer, the first intermediate failure region F_1 is then defined as

$$F_1 = \{\mathbf{x} : f(\mathbf{x}) > y_1^*\}, \tag{2.10}$$

where

$$y_1^* = \frac{y_0^{np} + y_0^{np+1}}{2}. \tag{2.11}$$

By construction, the samples $\mathbf{x}_0^1, ..., \mathbf{x}_0^{np}$ now lie within F_1, while $\mathbf{x}_0^{np+1}, ..., \mathbf{x}_0^n$ do not. Additionally, although any value of p within the bounds is acceptable, it was found by the authors of [52] that $p = [0.1, 0.3]$ is the most suitable choice to boost the efficiency of the algorithm. Using the direct MC approach, $P(F_1)$ can then be defined as

$$p(F_1) \approx \frac{1}{n}\sum_{i=1}^{n} I_{F_1}(x_0^{(i)}) = p. \tag{2.12}$$

This region F_1 can be considered as a conservative approximation to the true failure region F. Indeed, as $F \subset F_1$, p_F can be expressed as a product of probabilities:

$$p_F = P(F_1)P(F|F_1), \tag{2.13}$$

where $P(F|F_1)$ is the conditional probability of F given F_1. As $P(F_1)$ is known from Eq. 2.12, the task of estimating p_F is reduced to estimating this conditional probability. The next step is to populate the region F_1 by generating the remaining $(n - np)$ samples from the conditional distribution

$$\pi(\mathbf{x}|F_1) = \frac{\pi(\mathbf{x})I_{F_1}(\mathbf{x})}{P(F_1)} = \frac{I_{F_1}(\mathbf{x})}{P(F_1)}\prod_{i=1}^{d}\phi(x_i). \tag{2.14}$$

To achieve this, SuS employs Markov Chain Monte Carlo (MCMC) [53], which is a class of algorithms designed to sample from complex probability distributions. In particular, SuS utilises a particular MCMC algorithm known as the Modified-Metropolis Hastings (MMH) algorithm [54], which has been specifically developed for sampling from conditional probabilities such as $\pi(\mathbf{x}|F_1)$.

Given that $\mathbf{x} \sim \pi(\cdot|F_1)$ is a known sample from the conditional distribution $\pi(\cdot|F_1)$, the MMH algorithm utilises $\mathbf{x}$ to generate another sample $\tilde{\mathbf{x}}$ from $\pi(\cdot|F_1)$. This process is illustrated in figure 2.1. The algorithm starts by producing a candidate sample, $\xi = (\xi_1, ..., \xi_d)$, where for each input dimension, a prospective candidate value η_i is generated from a univariate proposal distribution that is centred at x_i, with the symmetric property $q_i(\eta_i \mid x_i) = q_i(x_i \mid \eta_i)$:

$$\eta_i \sim q_i(\cdot \mid x_i). \tag{2.15}$$

The acceptance ratio of the prospective candidate value is then evaluated

$$r_i = \frac{\phi(\eta_i)}{\phi(x_i)}, \tag{2.16}$$

and the actual i^{th} candidate value set accordingly:

$$\xi_i = \begin{cases} \eta_i & \text{with probability } \min\{1, r_i\}, \\ x_i & \text{with probability } 1\text{-}\min\{1, r_i\}. \end{cases} \tag{2.17}$$

Once all the candidate values are set, the candidate sample ξ is accepted or rejected it follows the conditional distribution $\pi(\cdot|F_1)$:

$$\tilde{\mathbf{x}} = \begin{cases} \xi & \text{if } \xi \in F_1, \\ \mathbf{x} & \text{if } \xi \notin F_1. \end{cases} \tag{2.18}$$

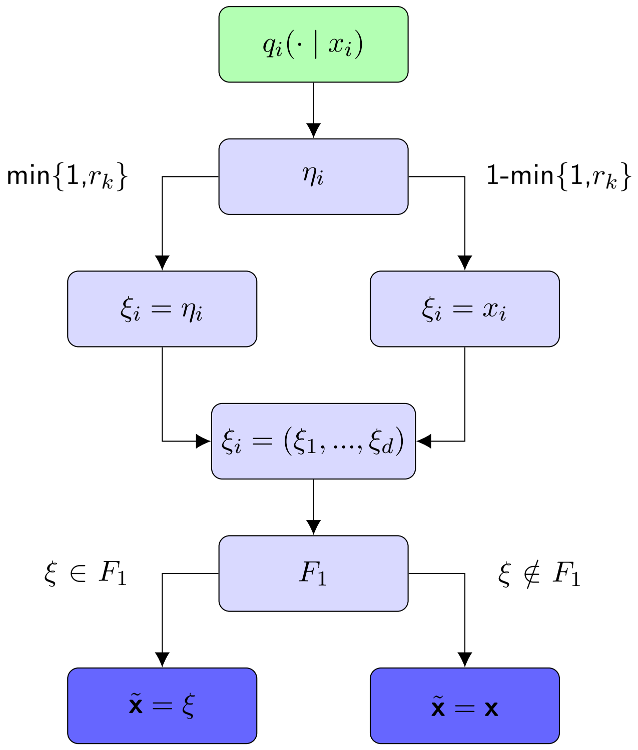


Figure 2.1: **Modified Metropolis-Hastings:** The steps involved in generating and accepting/rejecting a candidate sample using the MMH algorithm.

This way, $\tilde{\mathbf{x}}$ is guaranteed to follow the conditional distribution $\pi(\cdot|F_1)$ as required. In the case that the candidate sample ξ is rejected in Eq. 2.18, then $\tilde{\mathbf{x}} = \mathbf{x} \sim \pi(\cdot|F_1)$, and the requirement is automatically satisfied. Alternatively, in the case that ξ is accepted, then $\tilde{\mathbf{x}} = \xi$, and there is a distinct transition from $\mathbf{x}$ to $\tilde{\mathbf{x}} = \xi$. To prove that $\tilde{\mathbf{x}}$ follows the conditional distribution despite this transition, let $\zeta(\cdot)$ denote the PDF of $\tilde{\mathbf{x}}$, then

$$\zeta(\tilde{\mathbf{x}}) = \int_{F_1} \pi(\mathbf{x} \mid F_1) t(\tilde{\mathbf{x}} \mid \mathbf{x}) d\mathbf{x}, \tag{2.19}$$

where $t(\tilde{\mathbf{x}} \mid \mathbf{x})$ is the transition PDF from $\mathbf{x}$ to $\tilde{\mathbf{x}} \neq \mathbf{x}$. As the respective input variables of $\tilde{\mathbf{x}} = \xi$ are independently generated, the transition PDF $t(\tilde{\mathbf{x}} \mid \mathbf{x})$ can be expressed as a product:

$$t(\tilde{\mathbf{x}} \mid \mathbf{x}) = \prod_{i=1}^{d} t_i(\tilde{x_i} \mid x_i), \tag{2.20}$$

where $t_i(\tilde{x_i} \mid x_i)$ is the transition PDF for the i^{th} input variable $\tilde{x_i}$. Combining Eq. 2.14, Eq. 2.19 and Eq. 2.20 gives

$$\zeta(\tilde{x}) = \int_{F_1} \frac{I_{F_1}(\mathbf{x})}{P(F_1)} \prod_{i=1}^{d} \phi(x_i) \prod_{i=1}^{d} t_i(\tilde{x_i} \mid x_i) d\mathbf{x} = \frac{1}{P(F_1)} \int_{F_1} \prod_{i=1}^{d} \phi(x_i) t_i(\tilde{x_i} \mid x_i) d\mathbf{x}. \tag{2.21}$$

Central to proving that $f(\tilde{\mathbf{x}}) = \pi(\mathbf{x} \mid F_1)$ is to show that $\phi(x_i)$ and $t_i(\tilde{x_i} \mid x_i)$ satisfy the detailed balance equation:

$$\phi(x_i) t_i(\tilde{x_i} \mid x_i) = \phi(\tilde{x_i}) t_i(x_i \mid \tilde{x_i}) \tag{2.22}$$

The transition from $\mathbf{x}$ to $\tilde{\mathbf{x}}$ is determined not only by the proposal PDF $q_i(\tilde{x_i} \mid x_i)$, but also whether this proposal is actually accepted with probability $min\{1, r_i\}$. Therefore,

$$t_i(\tilde{x_i} \mid x_i) = q_i(\tilde{x_i} \mid x_i) \min\left\{1, \frac{\phi(\tilde{x_i})}{\phi(x_i)}\right\}, \; \tilde{x_i} \neq x_i. \tag{2.23}$$

By utilising the identity $a\min\{1, \frac{b}{a}\} = b\min\{1, \frac{a}{b}\}$ for any $a, b > 0$, the symmetric property of the proposal PDF $q_i(\tilde{x_i} \mid x_i) = q_i(x_i \mid \tilde{x_i})$ and Eq. 2.23, it can shown that

$$\begin{aligned} \phi(x_i) t_i(\tilde{x_i} \mid x_i) &= q_i(\tilde{x_i} \mid x_i) \phi(x_i) \min\left\{1, \frac{\phi(\tilde{x_i})}{\phi(x_i)}\right\} \\ &= q_i(x_i \mid \tilde{x_i}) \phi(\tilde{x_i}) \min\left\{1, \frac{\phi(x_i)}{\phi(\tilde{x_i})}\right\} = \phi(\tilde{x_i}) t_i(x_i \mid \tilde{x_i}), \end{aligned} \tag{2.24}$$

satisfying Eq. 2.22. As a result, combining Eq. 2.21, Eq. 2.22 and Eq. 2.23 gives

$$\begin{aligned}\zeta(\tilde{\mathbf{x}}) &= \frac{1}{P(F_1)} \int_{F_1} \prod_{i=1}^{d} \phi(\tilde{x}_i) t_i(x_i \mid \tilde{x}_i) d\mathbf{x} \\ &= \frac{1}{P(F_1)} \prod_{i=1}^{d} \phi(\tilde{x}_i) \int_{F_1} t(\mathbf{x} \mid \tilde{\mathbf{x}}) d\mathbf{x} = \pi(\tilde{\mathbf{x}} \mid F_1), \end{aligned} \tag{2.25}$$

since $t(\mathbf{x} \mid \tilde{\mathbf{x}})$ integrates to 1. Consequently, it is clear that the MMH algorithm produces samples according to the conditional distribution regardless of the acceptance/rejection of the candidate sample in Eq. 2.18. To generate the remaining $(n - np)$ samples from $\pi(\cdot \mid F_1)$, SuS uses the MMH algorithm to produce a succession of $(\frac{1}{p} - 1)$ new samples for each of the $\mathbf{x}_0^{(i)} \sim \pi(\cdot \mid F_1)$, $i = 1, ..., np$. For example, $\mathbf{x}_0^{(i)} = \mathbf{x}_{0,0}^{(i)} \rightarrow \mathbf{x}_{0,1}^{(i)} \rightarrow ... \rightarrow \mathbf{x}_{0,\frac{1}{p}-1}^{(i)}$, where for each $\mathbf{x}_{0,j}^{(i)}$, the previous sample $\mathbf{x}_{0,j-1}^{(i)}$ is used as the initial sample. The succession of samples is known as a Markov chain with the stationary distribution $\pi(\cdot \mid F_1)$ and the initial sample in the chain, e.g. $\mathbf{x}_{0,0}^{(i)}$, is referred to as the ‘seed’ of the Markov chain.

By generating an additional $\frac{1}{p} - 1$ for each $\mathbf{x}_0^{(1)}, ..., \mathbf{x}_0^{(np)}$, a total of n samples are produced that are located within the region F_1, and the region is said to be populated. For simplicity, these samples will be denoted $\mathbf{x}_1^{(1)}, ..., \mathbf{x}_1^{(n)}$, where the subscript 1 indicates that the sample lies within the first conditional level. Once these samples are generated, their respective performance values are computed and the samples rearranged and renumber accordingly: $y_1^{(1)} = f(\mathbf{x}_1^{(1)}) \geq ... \geq y_1^{(n)} = f(\mathbf{x}_1^{(1)})$. As F is often exceptionally small, it is unlikely that none of $\mathbf{x}_1^{(1)}, ..., \mathbf{x}_1^{(n)}$ lie within F, i.e. $y_1^{(i)} < y^*$ for all $i = 1, ..., n$, and further conditional levels need to be generated. Analogously to Eq. 2.10, the second intermediate failure region can be defined as

$$F_2 = \{\mathbf{x} : f(\mathbf{x}) > y_2^*\}, \tag{2.26}$$

where

$$y_2^* = \frac{y_1^{np} + y_1^{np+1}}{2}. \tag{2.27}$$

Since $y_1^{(i)} > y_1^*$ for all $i = 1, ..., n$, $y_2^* > y_1^*$ and $F \subset F_2 \subset F_1$. Thus, F_2 can be seen as a conservative approximation to F, but with greater accuracy than F_1. This can be seen clearly in figure 2.2, which illustrates the concept of SuS and showcases how the algorithm utilises MCMC to converge towards the true failure region.

As in the construction of F_1, by design $\mathbf{x}_1^{(1)}, ..., \mathbf{x}_1^{(np)}$ lie within F_2, while $\mathbf{x}_1^{(np+1)}, ..., \mathbf{x}_1^{(n)}$ do not. Consequently, the estimate of the conditional probability of F_2 given F_1 is equal to p,

$$P(F_2 \mid F_1) \approx \frac{1}{n} \sum I_{F_2}(x_1^{(i)}) = p. \tag{2.28}$$

Moreover, as $F \subset F_2 \subset F_1$, the conditional probability $P(F \mid F_1)$ from Eq. 2.13 can be expressed as a product:

$$P(F \mid F_1) = P(F_2 \mid F_1) P(F \mid F_2). \tag{2.29}$$

Combining Eq. 2.13 and Eq. 2.29 provides a new expression for the failure probability:

$$p_F = P(F_1) P(F_2 \mid F_1) P(F_2). \tag{2.30}$$

Taking Eq. 2.12 and Eq. 2.30 into account, the problem of estimating p_F is therefore reduced to estimating the conditional probability $P(F \mid F_2)$. The MMH algorithm can then used to populate F_2 by generating samples from $\pi(\cdot \mid F_2)$, before repeating the steps described above to define the third intermediate failure region $F_3 \subset F_2$. Using the same logic as before, the probability of failure can be defined $p_F = P(F_1)P(F_2 \mid F_1)P(F_3 \mid F_2)P(F \mid F_3)$, and the task of estimating it reduced to estimating $P(F \mid F_3)$. This process is repeated until F has been sufficiently sampled and the stopping criterion is met. Indeed, given that the number of samples that lie within the failure region F at conditional level L is defined as

$$n_F(L) = \sum_{i=1}^{n} I_F(x_L^i). \tag{2.31}$$

The stopping criterion can then be defined as

$$\frac{n_F(L)}{n} \geq p, \tag{2.32}$$

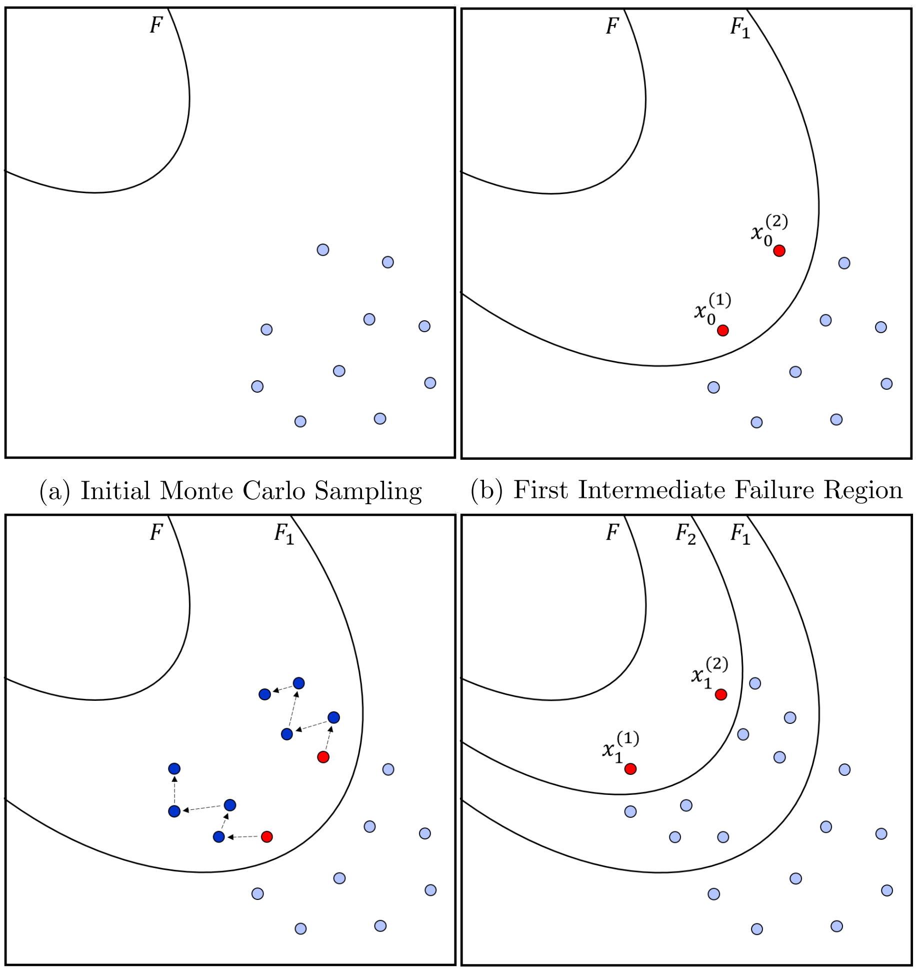
that is, at some level L with samples $x_L^{(1)}, ..., x_L^{(np)}$, at least np of them lie within the failure region. Once this is satisfied, the SuS algorithm terminates, with L as the last conditional level. Since F is a rare event, it is likely that $n_F(L) = 0$ for the first few conditional levels. However, as L increases, $n_F(L)$ will increase as F_L shrinks towards F. At conditional level L, the failure probability p_F is expressed as a product,

$$p_F = P(F_1)P(F_2 \mid F_1)...P(F_L \mid F_{L-1})P(F \mid F_L). \tag{2.33}$$

However, as both $P(F_1) = p$ and $P(F_i \mid F_{i-1})$=p for all $i = 2, ..., L$ by design, the probability of failure is defined as

$$p_F \approx p_F^{SuS} = p^L \frac{n_F(L)}{n}. \tag{2.34}$$

All distinct steps of the process described in this section are contained in a pseudocode located in the appendix as algorithm 4.



(a) Initial Monte Carlo Sampling (b) First Intermediate Failure Region

(c) MCMC Step (d) Second Intermediate Failure Region

Figure 2.2: **Subset Simulation Procedure** Graphical display of subset simulation. The light blue dots represent the initial MC samples in 2.2a, and samples from lower conditional levels in the other sub-figures. The red dots represent seed samples, and the blue dots represent samples produced via MCMC.

2.3 Single-Objective Optimisation using Subset Simulation

2.3.1 Analogy between Reliability and Optimisation

The previous section introduced the reliability problem, which is concerned with estimating the probability of failure of a system. This probability is associated with regions of the input space which correspond to an unacceptable system performance, as described in Eq. 2.1. The occurrence of unacceptable performance, defined by exceeding the critical threshold value y^*, is often exceedingly rare, which presents significant challenges when attempting to sample from the associated failure domain. This challenge provided the motivation behind the original SuS algorithm. It is fairly clear that altering y^* has a direct impact on the size of the failure domain: decreasing y^* will increase the size of the failure domain, whilst increasing y^* will do the opposite. In the latter case, given a certain value for y^*, only the input values corresponding to the most extreme system performance will lie within the failure domain. An optimisation problem consists of attempting to find the input values $\mathbf{x}_{opt}$ that maximises (or minimises depending on problem) the performance of a system, i.e. find $\mathbf{x}_{opt}$ such that $y_{opt} = f(\mathbf{x}_{opt})$. The domain for optimal values is often narrower than the failure domain in reliability problems, and given the correct context, optimisation can be seen as an extreme form of the reliability problem. Specifically, the task of optimising a system can be converted to a reliability problem in which the critical threshold is defined as $y^* = y_{opt}$ and the associated failure domain is simply $F = \mathbf{x}_{opt}$. This is illustrated for the one-dimensional case in figure 2.3. The function given by the blue line represents the performance values across the input domain. The green line represents a typical critical threshold for reliability problems y^*, with the solid green areas on the x axis showing the relative size of the associated failure domain. The solid red line is a more extreme critical threshold $y^* = y_{opt}$, that passes through the red dot that represents the optimal point of the function y_{opt}. The dashed red line shows the location of the failure domain associated with the solid red line.

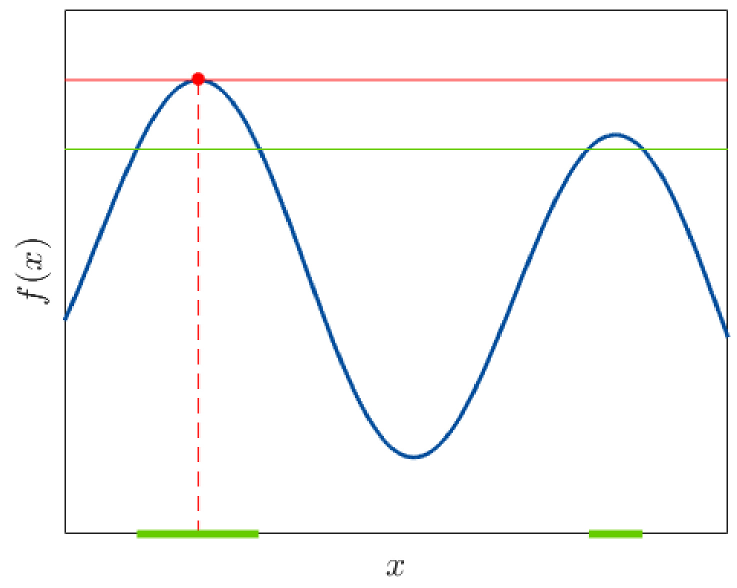


Figure 2.3: **Analogy between Reliability and Optimisation:** Similarities between optimisation and reliability allow the former to be framed in the context of the latter.

By definition, the probability of failure in the more extreme case is zero, however this is inconsequential as the actual aim is to locate the optimal input values. As highlighted in the previous section, SuS gradually converges towards a failure region, with each intermediate failure region F_i associated with an increased intermediate critical threshold compared to the previous critical threshold, i.e. $y_i^* \geq y_{i-1}^*$. Incidentally, as SuS proceeds, the performance values of accepted samples generally increases. Consequently, as SuS converges towards the extreme failure domain associated with the optimisation problem, it will generate samples with increasing performance, culminating in locating the true optimal input values [32, 55].

2.3.2 Procedure for Single-Objective Optimisation

Once the analogy between reliability and optimisation is made, it is fairly straightforward to apply SuS for the purposes of the latter. The task of single-objective optimisation can be described mathematically as

$$\begin{aligned} &\underset{x}{\text{Maximise}} && f(\mathbf{x}), \\ &\text{subject to} && g_j(\mathbf{x}) \geq 0, j = 1, 2, ..., J, \\ & && h_k(\mathbf{x}) = 0, k = 1, 2, ..., K. \end{aligned} \tag{2.35}$$

where g_j and h_k represent inequality and equality constraint functions for the given problem. As SuS frames the optimisation problem in the context of a reliability problem, each previously deterministic input variable is assigned a probability density function (PDF), denoted $\phi(x)$, to assist in guiding the search for promising solutions. It was found in [32] that a reasonable choice was to set these PDFs as truncated normal distributions, with mean μ as close as possible to the global optimal input values, as this can speed up the convergence towards optimal solutions compared to other choices, such as a uniform distribution. In the case that these optimal values are unknown, and there is no expert opinion, a suitable choice for the mean is at the midpoint of the input bounds. The standard deviation of the PDFs, σ, is defined as a function of the domain length L, usually $L = \sigma/6$. As a result, the PDF for the i^{th} input variable is defined as

$$\pi(x_i, \mu_i, \sigma_i, x_i^L, x_i^U) = \frac{\phi(\frac{x_i - \mu_i}{\sigma_i})}{\Phi(\frac{x_i^U - \mu_i}{\sigma_i}) - (\frac{x_i^L - \mu_i}{\sigma_i})}. \tag{2.36}$$

Here x_i^L, x_i^U are the respective lower and upper bounds for the given input variable, $\phi(\cdot)$ is the PDF and $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard Gaussian distribution. Additionally, the number of samples per level, n, and the level probability, p, are set.

The key stages of the process are shown in figure 2.4. Once again, the first step in the SuS algorithm is to generate n i.i.d samples $\{\mathbf{x}_0^1, ..., \mathbf{x}_0^n\}$ via MC sampling. The constraint values and performance values for each sample are then computed. The total constraint value for a given sample is defined as the number of individual constraints that it violates, rather than a measure of how much it violates a particular constraint. That is, for a given sample, the individual constraint violations are defined as

$$\nu_j(\mathbf{x}) = \begin{cases} 0 & \text{if } g_j(\mathbf{x}) \geq 0, \\ 1 & \text{if } g_j(\mathbf{x}) < 0. \end{cases} \qquad \gamma_k(\mathbf{x}) = \begin{cases} 0 & \text{if } h_k(\mathbf{x}) = 0, \\ 1 & \text{if } h_k(\mathbf{x}) \neq 0, \end{cases} \tag{2.37}$$

and the total constraint value is then defined as

$$f_{con}(\mathbf{x}) = \sum_{j=1}^{J} \nu_j(\mathbf{x}) + \sum_{k=1}^{K} \gamma_k(\mathbf{x}). \tag{2.38}$$

Samples are then sorted according to a double-criterion ranking method [56], where samples are first sorted and renumbered according to their constraint values

$$f_{con}(\mathbf{x}_0^1) \geq ... \geq f_{con}(\mathbf{x}_0^n). \tag{2.39}$$

Samples with the same constraint values are then sorted and renumbered according to their performance values. The first intermediate conditional level is therefore defined as

$$F_1 = \{\mathbf{x} : f(\mathbf{x}) > y_1^*, f_{con}(\mathbf{x}) \leq C_1^*\}, \tag{2.40}$$

where

$$y_1^* = \frac{y_0^{np} + y_0^{np+1}}{2}, \tag{2.41}$$

and

$$C_1^* = \frac{f_{con}(\mathbf{x}_0^{np}) + f_{con}(\mathbf{x}_0^{np+1})}{2}. \tag{2.42}$$

Populating F_1 and additional conditional levels follow the same MMH steps as discussed in section 2.2. A key component in MMH which dictates the overall performance of the SuS procedure is the proposal distributions $q_i(\cdot \mid \cdot)$. The authors in [47] observed that efficiency of the MMH algorithm was insensitive to type of the proposal PDFs (e.g. Gaussian, Uniform, etc), but heavily influenced by their spread (variance). A larger spread encourages exploration of the input space, however increases the likelihood of rejecting candidate samples. On the other hand, a smaller spread increases the likelihood of accepting candidate samples, but encourages clustering of samples to local areas in the input space, which may omit other areas with superior performance. It was found in [52] that the optimal acceptance rate for the MMH algorithm is between 0.3 and 0.5. To maintain an acceptance rate within this bound, the spread of the proposal PDFs has to be constantly adapted. For the implementation in this thesis, the proposal PDFs were chosen to be uniformly distributed with spread $\tilde{\sigma}$. This spread was adapted to try and obtain an acceptance rate of 0.4 at each iteration, where an iteration consisted of a single sample from each Markov chain. That is, for a given spread at iteration i, the spread at the next iteration is defined as

$$\tilde{\sigma}_{i+1} = \tilde{\sigma}_i\big(1 - (0.4 - A_i)\big), \tag{2.43}$$

where A is the acceptance rate from iteration i.

One of the major changes adapting SuS for optimisation is with regards to the stopping criterion. Reliability problems begin with a predefined critical threshold y^*, which is used to inform a suitable stopping criterion (Eq. 2.32). In optimisation problems, such a threshold is rarely known beforehand, and as such, an alternative stopping criterion is required. The most simple solution is to assign a finite number of computational model evaluations, or computational budget, for the given problem. The algorithm would proceed until this budget is exhausted and the best performing sample taken as the estimate for the global optimal. This is particularly useful when working with computationally expensive models that may limit the number of evaluations. However, in the case that the algorithm locates the optimal values before exhausting the computational budget, there would be no means to stop the algorithm from wasting unnecessary resources. A solution to address this issue is to use the adaptive spread of the proposal distribution as a measure of the convergence of the overall algorithm. In general, as the algorithm proceeds, this spread will decrease as it becomes increasingly difficult to generate samples that exceed the required performance threshold to be accepted. Once the spread has shrunk below a certain threshold, it is likely that there will be limited to no improvement in the estimated global optimal by continued sampling, and the SuS algorithm can be said to have essential converged to its final solution. As a result, a stopping criterion for the optimisation case can be defined as the point where

$$\tilde{\sigma} \leq \epsilon, \tag{2.44}$$

for some predefined threshold ϵ. This stopping criterion however takes no account into the computational costs involved in sampling until this condition is met, and can be infeasible in problems with significant computational constraints. A solution is to employ both of the stopping criterion mentioned above, and stopping the algorithm once a single one of them is satisfied. This way, needless computational expense is wasted in the case that the SuS algorithm converges early, and alternatively the algorithm successfully stops in the case it reaches its allotted model evaluations.

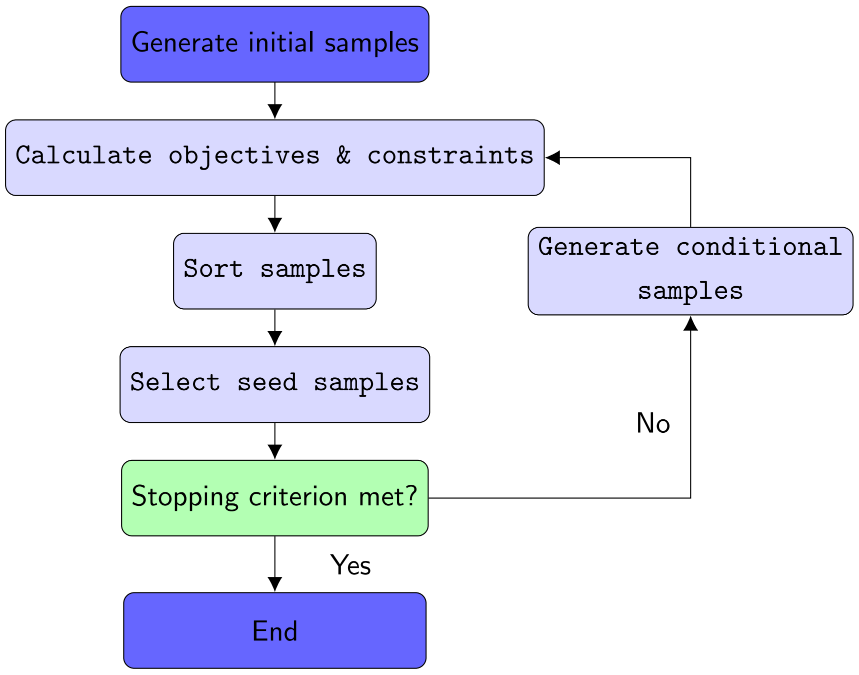


Figure 2.4: **Single-Objective Optimisation using Subset Simulation:** The key steps involved in the optimisation procedure.

2.3.3 Single-Objective Optimisation Test Problem

In this section, an illustrative test problem is used to showcase the steps described in section 2.3.2. The code for all SuS-based algorithms utilised in this research were written and developed within MATLAB, without the use of any existing toolboxes. This not only encouraged a greater understanding of the various theoretical components of the algorithm, but on a practical level, it provided a higher level of flexibility to be able to adapt the algorithm in order to maximise performance for a variety of problems.

The test problem is known as the Schwefel function and is defined as

$$\begin{aligned} &\underset{x}{\text{Minimise}} && f(\mathbf{x}) = 418.9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{|x_i|}), \\ &\text{subject to} && x_i \in \big[-500, 500\big], i = 1, ..., d, \end{aligned} \tag{2.45}$$

where d is the number of input dimensions. The Schwefel function was selected as a test problem as it contains many local optima, which presents a reasonable challenge for optimisation algorithms to successfully converge to the true global optimal point. Additionally, the problem was set to be two-dimensional, i.e. $d = 2$, as then the optimisation procedure can be visualised and the results more clearly understood. The SuS algorithm was applied to the problem with $p = 0.2$, 200 samples per condition level and a maximum budget of 1500 evaluations. Figure 2.5a presents the output space of the Schwefel function in the form of a contour plot, as well as an accompanying colour bar. On inspection, there is clearly several areas possessing dark blue colouring, primarily in the four corners of the plot. These areas correspond to local minima, and as such are of interest in this problem. Figure 2.5b contains the samples produced during the SuS procedure. The blue dots represent the initial samples produced via MC, evident by the wide distribution of samples across the entire domain. The orange dots represent samples from the first conditional level, which are more restricted in their distribution, clearly located in areas corresponding to a low output value (i.e. areas tending towards the bottom of the colour bar). The yellow dots represent the second conditional level, which are entirely contained within proximity of the optimal in the top right of the contour plot. Note how these samples are significantly less spaced than the samples from previous levels, as the spread of the proposal distribution (Eq. 2.43) is reduced as the threshold becomes stricter. Finally the red star represents the optimal solution according to the SuS algorithm, located at $\mathbf{x} = [420.9687, 420.9687]$, which is the domain of the global optimal of the problem.

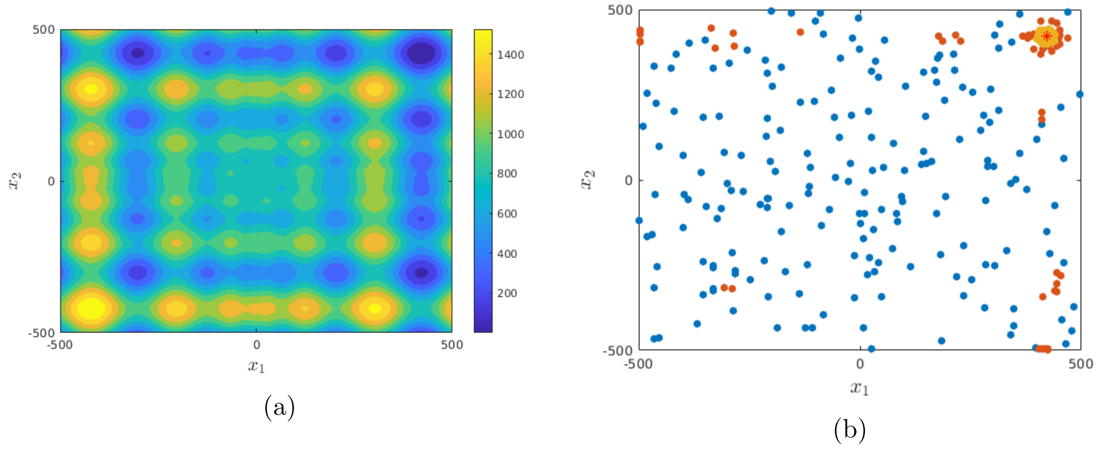


Figure 2.5: **Schwefel Problem:** (a) depicts a contour plot of the function for the input domain. (b) shows samples from several conditional levels within the SuS algorithm.

The motivation behind the test problem was to visibly showcase the main concepts of the SuS algorithm. The combination of the two figures highlights how samples converge towards areas of promising performance, in this case areas with low output values.

2.4 Multi-Objective Optimisation using Subset Simulation

In most real-world problems, there are often multiple objectives to consider simultaneously. Multi-objective optimisation problems (MOPs) involve attempting to optimise these objectives, which are often in conflict with one another. As a result, rather than a single global optimal solution, MOPs possess a set of solutions known as the Pareto front [27]. The mathematical definition of an MOP is

$$\begin{aligned} &\underset{x}{\text{Minimise}} && \Big\{f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_M(\mathbf{x})\Big\}, \\ &\text{subject to} && g_j(\mathbf{x}) \geq 0, j = 1, 2, ..., J, \\ & && h_k(\mathbf{x}) = 0, k = 1, 2, ..., K, \end{aligned} \tag{2.46}$$

where $\mathbf{x}$ are the input variables, f_i represent the objective functions, while g_j and h_k represent inequality and equality constraints. Despite the added complexity of the needing to consider multiple objectives, it was shown that the SuS procedure remains broadly the same as in the single-objective case [57]. The input variables are assigned PDFs and the initial samples $\mathbf{x}_0^{(1)}, ..., \mathbf{x}_0^{(n)}$ generated once more via MC. The corresponding performance and constraint values are then computed, with the latter done as in Eq. 2.38. As in the single-objective case, the samples then need to be sorted, however as there is now more than one objective value, this step requires additional consideration. The Non-Dominated sorting algorithm [58] is employed to sort the samples; the algorithm takes two parameters, nondomination rank and crowding distance, to compare and order samples.

2.4.1 Non-Dominated Sorting

The nondomination rank compares the superiority of one samples to another. Given two samples, X_1, $X_2 \in \mathbb{R}^d$:

1. If X_1 is feasible and X_2 is infeasible (violates any constraints in Eq. 2.46), then X_1 dominates X_2, and vice versa.
2. If both X_1 and X_2 are feasible, and none of the objective values associated with X_1 are worse than those associated with X_2, and at least one is better, then X_1 dominates X_2, and vice versa.
3. If both X_1 and X_2 are infeasible, and if X_1 violates less constraints than X_2, then X_1 dominates X_2, and vice versa.
4. If none of the above are satisfied, X_1 and X_2 are non-dominated to one another.

Using the non-domination rank, samples are sorted into numbered groups known as fronts; all samples in a front are non-dominated to each other. Any sample in a group with a higher nondomination-rank should be dominated by at least one sample in a front with a lower nondomination-rank.

Once the samples are sorted into their respective fronts, the samples within a particular front are reordered with respect to their crowding distance value, which denotes the proximity of a sample to other samples in the objective space. Given a problem with M objective functions, for each objective function f_k, $k = 1, ..., M$, the samples within the front are reordered from largest to smallest according to their respective values for f_k. These values are then transformed to $\tilde{f}_k(0 \leq \tilde{f}_k \leq 1)$ by linear normalization. Then the crowding distance for some sample X_i with respect to the objective function f_k is defined

$$C_{ik} = \tilde{f}_k(i+1) - \tilde{f}_k(i-1),$$

that is the distance between the samples two nearest neighbours. The two end samples, i.e. samples with largest and smallest values of $\tilde{f}_k$, are assigned $C_{ik} = 1$ to encourage these samples to have a larger crowding distance value, and ensure sample diversity. After the process is completed for all objective functions, the total crowding he distance value for a sample X_i is

$$C_i = \sum_{j=1}^{m} C_{ij}.$$

The crowding distance is shown visually in figure 2.6. Once the total crowding distance values are computed for all the samples in a front, the samples are reordered according to these values. Thus, the samples are now completely ordered and the selection of seeds can take place.

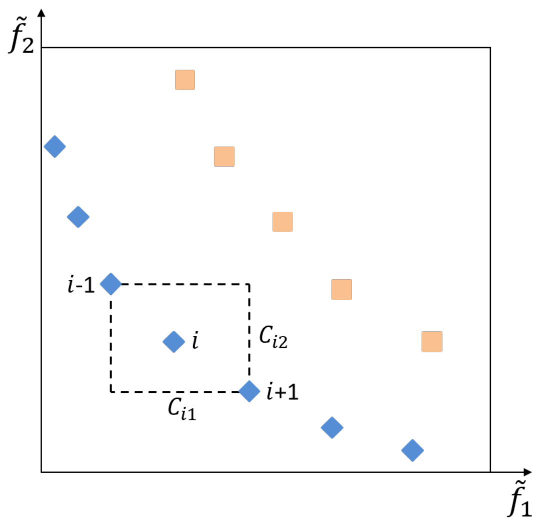


Figure 2.6: **Crowding Distance:** The diamonds represent samples from a front with a lower nondomination rank than the front consisting of the squares. An example of how crowding distance is found is shown by the rectangle with sides C_{i1} and C_{i2}.

Once the samples have been sorted, a proportion p of the best performing samples are selected as seeds to populate the next conditional level via MCMC. To increase diversity of samples and boost performance, a reordering strategy is also included, depicted in figure 2.7. Given the nature of the sorting, the original set of seed samples S_O may contain repeated samples. That is, given that S_O is made up of $n_s = np$ seeds, it contains n_u unique seeds, where $n_u \leq n_s$. Removing duplicated samples leaves a new set, S_U, consisting of these unique seeds which follow the same order as before. Next a final set, S_R, is constructed by continually adding the ordered samples from S_U until S_R contains n_s samples. The set S_R is then taken as the reordered set of seed samples. If the stopping criterion, usually defined by computational budget or a convergence measure, is met, then these seeds are taken to be the solution to the MOP. If the stopping criterion is not met, the algorithm proceeds to the next conditional level. Populating the next conditional level broadly follows the same steps as the single-objective case; MMH is used to produce candidates from seed samples which are accepted based on exceeding a certain threshold of performance. However, as there are multiple objectives to consider, it is not possible to easily define a performance threshold in the same manner as the single-objective case. Instead, a candidate sample is accepted if it lies within the current conditional level, defined by whether it is non-dominated to a set of q threshold samples, where $q \in [1, np]$. These threshold samples are randomly selected seed samples, and the value for q can be tuned to favour exploration or exploitation as desired. Consequently, an accepted candidate sample must improve or at least supplement the current Pareto front. This process is repeated until the chain is complete. As the seeds follow the conditional distribution, by design any accepted samples also follow the desired conditional distribution. Thus, a new conditional level is populated. The stages of the algorithm are contained in figure 2.8.

2.4.2 Multi-Objective Optimisation Test Problem

Similar to the single-objective case, an illustrative test problem is presented in this section to showcase the SuS algorithm being applied to a multi-objective optimisation problem. The majority of the code is identical to the code employed to perform single-objective optimisation, with the only changes present to incorporate non-dominated sorting and the reordering strategy.

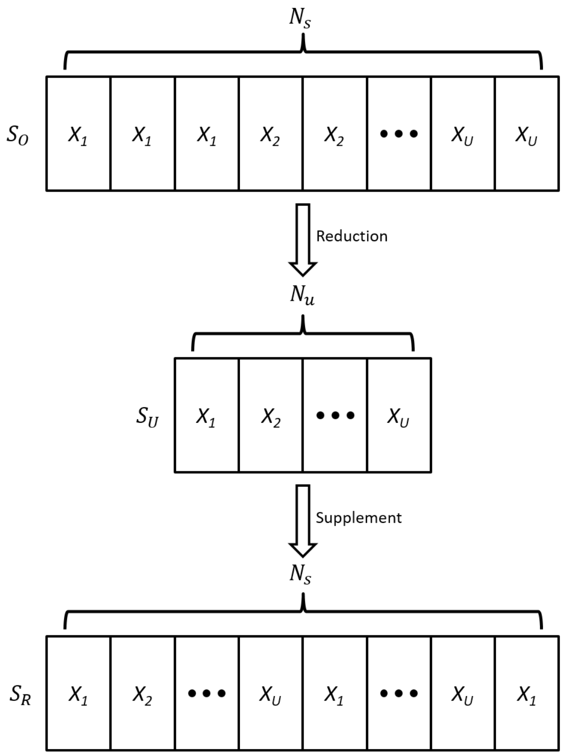

N_s
S_O
X_1 X_1 X_1 X_2 X_2 ••• X_U X_U
Reduction
N_u
S_U
X_1 X_2 ••• X_U
Supplement
N_s
S_R
X_1 X_2 ••• X_U X_1 ••• X_U X_1

Figure 2.7: **Reordering Strategy:** Graphical display of the reordering strategy in practice.

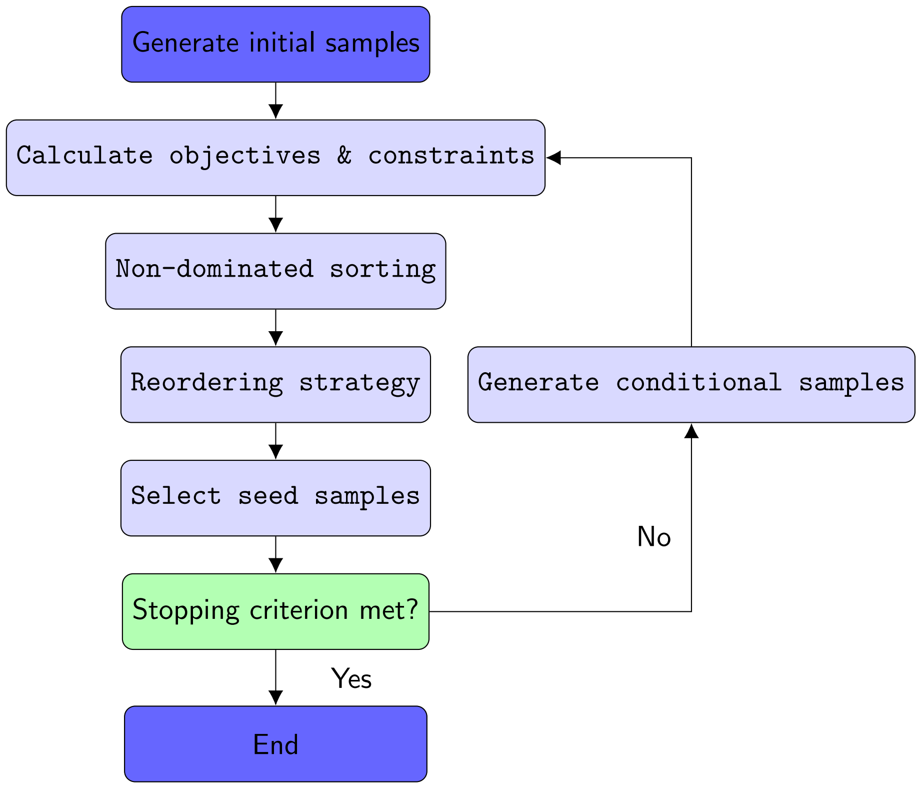


Figure 2.8: **Multi-Objective Optimisation using Subset Simulation:** The key steps involved in the optimisation procedure.

The Osyczka1 problem [59] is utilised as the test problem and is defined as

$$\begin{aligned} \underset{\mathbf{x}}{\text{Minimise}} \quad & \begin{cases} f_1(x_1, x_2) & = x_1 + x_2^2, \\ f_2(x_1, x_2) & = x_1^2 + x_2, \end{cases}, \\ \text{subject to} \quad & 2 \le x_1 \le 7,\ 5 \le x_2 \le 10, \\ & x_1 + x_2 - 12 \ge 0, \\ & x_1^2 - x_2^2 + 10x_1 + 16x_2 - 80 \ge 0. \end{aligned} \tag{2.47}$$

The Osyczka1 problem is a popular test problem for multi-objective optimisation algorithms, and was selected as it is a two-dimensional problem which possesses suitable difficulty but also the ability to visually observe how the SuS algorithm tackles optimising multiple objectives whilst balancing added constraints. The SuS algorithm was applied to the problem with $p = 0.2$, 500 samples per condition level and a maximum budget of 5000 evaluations. Figure 2.9a presents the input values of samples generated during the SuS procedure. The blue dots represent the initial samples produced via MC that are feasible, i.e. that satisfy the constraint functions. Any samples that violated the constraints are shown by grey dots. On inspection, there is a rough border from the top left of the plot to the bottom right which separates the feasible input domain from the infeasible input domain. Samples from the first and second conditional levels are given by the orange and yellow dots respectively. As in the single-objective test problem, the density of samples increases as the conditional level increases and the algorithm converges towards optimal regions. Finally, the last conditional level is given by the red dots, which act as the final solutions to the problem. Unlike in the single-objective case, there is no one global solution, but a series of solutions, known as Pareto optimals, which showcase the trade-off between the two objective functions. Figure 2.9b presents the corresponding objective values to the inputs provided in Figure 2.9a, with the same colour scheme used for clarity. Figure 2.9c provides a clearer view of the final Pareto front for the problem, and is as expected for this given problem. The goal of the test problem was to highlight that, with only minor adaptation, the SuS algorithm has the ability to efficiently tackle problems with multiple objectives.

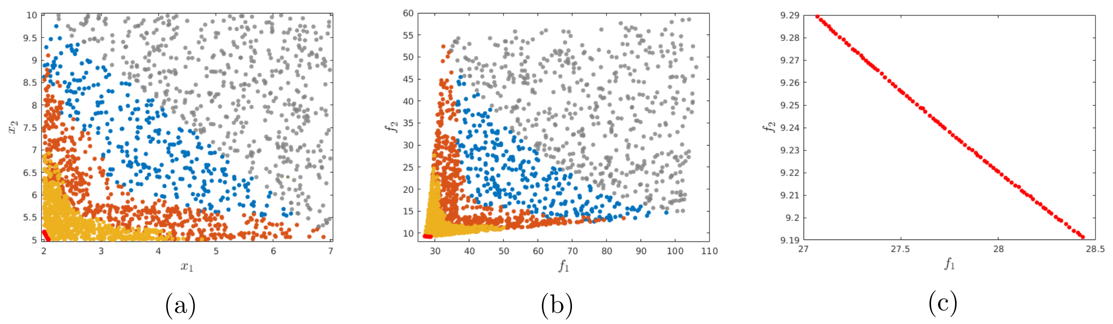


Figure 2.9: **Osyczka1 Problem:** (a) depicts input values of samples from several conditional levels within the SS procedure, (b) provides the corresponding objective values, and (c) exhibits the final Pareto front.

2.5 Concluding Remarks of Chapter

This chapter presented a thorough overview of subset simulation. The result is an optimisation algorithm that is sufficiently flexible to tackle a wide class of optimisation problems, including high-dimensional problems with extremely small inputs domains corresponding to areas of high performance. Foundational texts on subset simulation can be found in [47, 49] for reliability and [32, 55, 57] for optimisation respectively.

The optimisation algorithms presented within this chapter are designed for purposes of nominal optimisation. As such, they are not necessarily directly suitable for robust optimisation, which is of importance within this dissertation. However, they do provide the foundation for the novel method discussed in chapter 3, which addresses these concerns.

Chapter 3

Multi-Fidelity Robust Subset Simulation[1]

This chapter presents a novel approach to perform efficient robust optimisation of computationally expensive models. The method, termed robust subset simulation (RSS) extends the subset simulation algorithm described in chapter 2 to factor input uncertainty into the optimisation process. The chapter begins by providing a mathematical description of robust optimisation problems. Next, a brief overview of subset simulation for optimisation is presented for the benefit of the reader. The various steps involved to extend subset simulation to robust subset simulation are then covered in detail. Further, the method is generalised to factor in multiple levels of fidelity, denoted MF-RSS, to further reduce computational cost. Finally, two illustrative examples are presented to showcase the merits of the method, before it is applied to an industrial case study.

3.1 Introduction

A key aspect in engineering is the process of selecting a design that satisfies several constraints and performance objectives simultaneously. Modern engineering products often have to balance performance against factors such as profitability and environmental impact. There are often many feasible designs that satisfy these requirements. Identifying

[1]The results and ideas discussed in this chapter have been submitted for publication as a manuscript, see [60].

such designs, and subsequently selecting an optimal choice, is often a challenging task. Traditionally, this task has been interpreted as a multi-objective optimisation problem (MOP), defined mathematically as

$$\begin{aligned} &\underset{x}{\text{Minimise}} && \Big\{f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_M(\mathbf{x})\Big\}, \\ &\text{subject to} && g_j(\mathbf{x}) \geq 0, j = 1, 2, ..., J, \\ & && h_k(\mathbf{x}) = 0, k = 1, 2, ..., K. \end{aligned} \tag{3.1}$$

Here, $\mathbf{x}$ are the input variables, f_i represent the objective functions, g_j and h_k represent inequality and equality constraints. It is unusual for a single design to optimise each objective, and as such most MOPs possess a set of optimal solutions known as the Pareto front [27]. A variety of methods have been developed to tackle MOPs, such as Genetic Algorithms (GA) [30], Particle Swarm Optimisation (PSO) [31], Simulated Annealing [61] and Ant Colony Optimisation (ACO) [62]. Typically in industrial applications, the explicit form of the objective functions are unknown, for example computational fluid dynamics (CFD) models [15] and finite element models [63]. Such models are often computationally expensive, and limit the applicability of the aforementioned optimisation methods. A common solution to tackle this problem is to utilise a surrogate model to approximate the objective values. The optimisation methods can then be applied to this surrogate in place of the computationally expensive model, whilst satisfying the computational constraints. However, surrogate models introduce uncertainty, as they are only an approximation of the computational models. Thus, user input is also often required, with might be undesirable. Another option is to extend the direct optimisation methods to utilise multi-fidelity (MF) data, as done with GA [64]. Such approaches are applicable when more than one potential computational model exists for the system under study. Lower-fidelity (LF) models are defined by a lower-computational cost, but lower accuracy, than higher-fidelity (HF) models. For example, a LF model may employ a more simplistic mathematical model than a HF model, or in the case of employing the same model, do so with a mesh with a lower resolution. The core assumption in this chapter is that the optimisation methods can be applied to the LF model, and infer areas of interest for the HF model with a much lower overall computational expense than working solely with the HF model.

The goal of traditional optimisation methods is to locate the global optima of the underlying objective function. However, such optima can often be sensitive to changes in their inputs or environment [129]. Given the complex nature of modern engineering systems, uncertainties often arise during the manufacturing and life cycle of a design. Consequently, it is often more appropriate to seek solutions with high performance that are insensitive, or robust, to such uncertainties [65]. This is known as a robust multi-objective optimisation problem (RMOP). The effects of uncertainty can be generally categorised into three scenarios. The first introduces perturbations on the input variables [66], for example degradation during the life cycle altering the geometry. The second concerns noise affecting the objective evaluations [67], such as imprecise sensor measurements. The final form of uncertainty is the situation where the definition of the MOP itself changes over time, due to varying operational/environmental conditions [68]. These three sources of uncertainty and overall RMOP can be summarised mathematically as

$$\begin{aligned} \underset{x}{\text{Minimise}} \quad & \big\{f_1(\mathbf{x}_\epsilon, \mathbf{e}_1) + \nu_1, f_2(\mathbf{x}_\epsilon, \mathbf{e}_2) + \nu_2, ..., f_M(\mathbf{x}_\epsilon, \mathbf{e}_M) + \nu_M\big\}, \\ \text{subject to} \quad & g_j(\mathbf{x}_\epsilon) \geq 0, j = 1, 2, ..., J, \\ & h_k(\mathbf{x}_\epsilon) = 0, k = 1, 2, ..., K. \end{aligned} \tag{3.2}$$

Here, $\mathbf{x}_\epsilon$ represents a neighbourhood of input variables, defined by some value $\epsilon > 0$ to factor in perturbation within the input variables, i.e. $\mathbf{x} \in \big[\mathbf{x} - \epsilon, \mathbf{x} + \epsilon\big]$. The noise in the i^{th} objective function is represented by ν_i whilst the environmental conditions are contained in the vector $\mathbf{e}_i$. Finally, $g_j(\mathbf{x}_\epsilon)$ and $h_k(\mathbf{x}_\epsilon)$ are the respective inequality and equality constraints of the neighbourhood. This chapter deals with RMOPs where the only uncertainty lies within the perturbations of the input variables. As such, the goal is to locate designs whose performance is insensitive to alterations in its inputs. In the literature, several traditional optimisation methods have been extended to address this problem. The authors of [69] combine a GA to optimise the performance of the neighbourhood of $\mathbf{x}$, which is estimated using local approximation models. Another approach presented in [70] constructs an effective performance value, based on averaging the performance of a number of samples within the neighbourhood. A GA can then be used to optimise this effective value, or optimise the original performance subject to the effective value remaining above a certain threshold. Rather than the averaged performance, some approaches attempt to optimise the worst-case scenario; [71] employ a co-evolutionary GA while [72] opt for an enhanced PSO algorithm. A common problem with many methods proposed in the current literature is the absence of realistic computational budgets, and as such, the applicability of the methods to problems of industrial scale. One approach that does consider this problem is discussed in [73], which combines importance sampling and an archive of all solutions to attempt to reduce the computational expense of the optimisation process.

Factoring input uncertainty into the optimisation process can have a profound impact on the domain of optimal solutions. Rather than solely considering the performance associated with a single point, the process also has to factor in the performance of the region surrounding the point. Consequently, the goal of the robust optimisation procedure is primarily locating small regions of the input space that possess high performance across the entire region. This is not dissimilar to the reliability problem, which involves locating regions of the input space that correspond to performance values that exceed some defined critical threshold. As discussed in the previous chapter, an approach that has had great success in tackling the reliability problem is subset simulation [47]. The similarity between the task of reliability and robust optimisation, alongside the ability of SuS to sample from extremely small regions of high-dimensional input domains, makes the algorithm a suitable candidate to be applied for the purposes of robust optimisation.

The work in this chapter extends the subset simulation algorithm to factor in input uncertainty to perform efficient robust design within a realistic computational budget. Additionally, the method is generalised to be able to take advantage of multiple levels of fidelity. The chapter is organised as follows. Section 3.2 provides a brief overview of SuS for optimisation. The proposed method is introduced in Section 3.3 and its main components discussed. Two illustrative examples and an industrial CFD case study are presented in Section 3.4. The final section provides relevant conclusions and highlights future work.

3.2 Optimisation using Subset Simulation

The key steps in the procedure for employing SuS to tackle optimisation problems are described in detail in chapter 2. For the readers benefit, the SuS procedure for multi-objective optimisation is summarised in the pseudocode presented below in Algorithm 1.

Algorithm 1 Subset Simulation for Multi-objective Optimisation

```
1: Set $p$, $n$.
2: $N_s$=$np$
3: $N_{states}$=$\frac{1-p}{p}$
4: Generate $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ via MC
5: Evaluate objective and constraint values $\mathbf{Y} = f_1(\mathbf{X}), ..., f_M(\mathbf{X})$, $g_j(\mathbf{X})$, $h_k(\mathbf{X})$
6: While Stopping criterion not met
7:     Sort $\mathbf{X}, \mathbf{Y}$ using non-dominated sorting algorithm
8:     Set $N_s$ best samples as seeds.
9:     Randomly select one seed sample to act as threshold
10:    MCMC to produce candidate samples
11:    Compare candidates to threshold sample using non-dominated sorting
12:    Accept samples that are non-dominated to or dominate threshold, reject others.
13:    Repopulate $\mathbf{X}$ using accepted MCMC samples
14:    Evaluate $\mathbf{Y}$, $h_j(\mathbf{X})$, $w_k(\mathbf{X})$
15:    Adapt spread of proposal distribution
16: End
17: Sort $\mathbf{X}, \mathbf{Y}$ using non-dominated sorting algorithm
18: Select best samples as Pareto solutions.
```

3.3 Proposed Approach

The goal of the proposed approach is to perform efficient robust optimisation of computationally expensive models. The method extends the multi-objective SuS algorithm to factor in input uncertainty, denoted Robust Subset Simulation (RSS). Moreover, the RSS is generalised to multiple levels of fidelity and termed Multi-Fidelity Robust Subset Simulation (MF-RSS). The method attempts to address the short-comings of both traditional optimisation methods and robust optimisation methods in their application to computationally constrained problems. Further details of approach are discussed in the following subsections.

3.3.1 Robust Optimisation using Subset Simulation

To factor input uncertainty into the SuS algorithm described in section 3.2, the proposed approach utilises the effective objective value and type I robustness definitions from [70].

The effective objective value is defined as

$$f^e(\mathbf{x}) = \frac{1}{|\mathbf{x}_\epsilon|} \int_{\mathbf{x}^* \in \mathbf{x}_\epsilon} f(\mathbf{x}^*)\ d\mathbf{x}^*, \tag{3.3}$$

where $\mathbf{x}_\epsilon$ is the neighbourhood of samples around $\mathbf{x}$ and $|\mathbf{x}_\epsilon|$ its hypervolume. The RMOP using type I robustness is defined as the MOP defined in Eq. 3.1 but with the effective objective values as opposed to the original values, and constraint functions applied over the neighbourhood, that is:

$$\begin{aligned} &\underset{x}{\text{Minimise}} && \left\{f_1^e(\mathbf{x}), f_2^e(\mathbf{x}), ..., f_M^e(\mathbf{x})\right\}, \\ &\text{subject to} && g_j(\mathbf{x}_\epsilon) \geq 0, j = 1, 2, ..., J, \\ & && h_k(\mathbf{x}_\epsilon) = 0, k = 1, 2, ..., K. \end{aligned} \tag{3.4}$$

The integral in (3.3) is usually approximated using N_ϵ neighbourhood samples via Monte carlo:

$$f^e(\mathbf{x}) \approx \frac{1}{N_\epsilon} \sum_{j=1}^{N_\epsilon} f(\mathbf{x}_j^*). \tag{3.5}$$

Most approaches in the literature that employ a form of averaged performance begin with a set value for N_ϵ and evaluate N_ϵ neighbourhood samples for every single sample in the optimisation procedure. This is computationally inefficient, and infeasible for most real-world problems. Indeed, computational inefficiency and the balance between performance and robustness are the main issues in current robust optimisation methods. The proposed method, RSS, considers several strategies to address these issues.

Initial Optimisation Stage

The first strategy is to recognise that one of the goals of robust optimisation is to locate designs with near optimal performance. Given a total computational budget of E evaluations, RSS begins by splitting E into a budget for a nominal optimisation stage, E_N, and a budget for a robust optimisation stage, E_R. The former is used to first probe the input space for regions with above-average performance, without considering input perturbation. By performing a brief optimisation stage at the beginning of the robust optimisation procedure, areas of low performance can be discarded, avoiding unnecessary computational waste. This optimisation is performed using the SuS algorithm following the steps described in Algorithm 1.

To maximise the efficiency of this initial optimisation, special care is given to the generation of candidate samples during the MCMC portion of the algorithm, described in section 2.2. In particular, the authors of [47] noted that the spread of the proposal distribution, rather than the type of distribution, was more influential in the overall efficiency of the MCMC process. According to [52], the optimal acceptance rate of samples is 0.4, as this ensures sufficient convergence whilst discouraging clustering of samples. To achieve this, the spread of the proposal distribution, $\tilde{\sigma}$, is continuously adapted over each state of the total number of Markov chains (n_s):

$$\tilde{\sigma}_{i+1} = \tilde{\sigma}_i \big(1 - (0.4 - A_i)\big).$$

Here A_i is the acceptance rate of samples at iteration i. Essentially, if this value is greater than the optimal acceptance rate, the spread for the next set of chains will be larger as to encourage exploration of samples, and vice versa. Additionally, as the samples should converge at each increasing level, the initial value of $\tilde{\sigma}$ is set to half of the final value of the previous conditional level. Specifically, the work in this chapter employs a uniform proposal distribution as it satisfies the requirement of being symmetric, and the impact of the adaptation of $\tilde{\sigma}$ is easily interpretable. Once this initial optimisation stage is complete, a proportion of the best performing samples, denoted X_{Nom}, are then used as the initial seed samples for the robust optimisation stage. Here the subscript Nom refers to the fact these samples are the 'best' according to nominal optimisation. Additionally, all evaluated samples and their respective objective values are stored in a bank of solutions, B, similar to the *archive* discussed in [73], to be used in the next stage.

Robust Optimisation Stage

The robust optimisation stage of RSS begins by setting an initial threshold of neighbourhood samples required for a sample to proceed to the first conditional SuS level, denoted $N_T = 1$. The seed samples provided from the initial optimisation stage are then compared with B to check how many neighbourhood samples they each possess, denoted N_ϵ. A further $N_T - N_\epsilon$ neighbourhood samples are generated for any seed samples which fail to reach this threshold, with all newly generated samples added to B. This allows effective values to be computed for all seed samples according to Eq. (3.5). A seed sample is then chosen at random to be the threshold sample, X_T, for the conditional level. The effective objective values of the threshold sample are used to measure the suitability of candidate samples. A candidate sample, X_{Cand} is generated in the same manner as SuS, however, unlike other robust optimisation procedures, no additional candidate neighbourhood samples are computed. Next, X_{Cand} is compared with B to check if it already possesses neighbourhood samples. If it does, the objective values of X_{Cand} are altered to effective objective values, which are then compared with the effective values of X_T. If it does not, the original objective values are used instead. This is to avoid the computational expense of computing $N_T - N_\epsilon$ neighbourhood samples per candidate, whilst still maintaining only candidates with reasonable promise are accepted. At any time a new sample is computed, it is added to the bank of solutions. Once the conditional level is populated, all samples are checked to see whether they possess at least N_T neighbourhood samples. Those that satisfy this requirement are placed into a set S_T while those with insufficient neighbourhood samples are placed into a set S_I. The latter set then undergoes non-dominated sorting, and $N_T - N_\epsilon$ neighbourhood samples generated for the n_s best performing samples. This ensures these samples each have at least N_T neighbourhood samples, and they are subsequently added to S_T. The newly supplemented S_T is then sorted and the best n_s samples selected as seeds for the next conditional level. At this point, the number of samples per conditional level, N, is reduced, to encourage exploitation over exploration as the algorithm proceeds. Additionally, the neighbourhood samples threshold is increased to $N_T = min(Ld, M)$, where L is the current conditional level and M is the maximum threshold value, to gradually increase the influence robustness has on the optimisation procedure. The value of M can be tuned depending on the goals of the user. In this chapter, $M = 5d$ was chosen to balance the validity of the robustness of solutions, against the associated computational strain. The algorithm proceeds until the required budget for the next conditional level exceeds the evaluations remaining in E_R. At which point, the best performing samples that exceed the current N_T are selected as the robust solutions to the problem, denoted X_{RS}.

3.3.2 Multi-Fidelity Robust Subset Simulation

The reasoning behind the original SuS algorithm is essentially given a sample performs well according to some criterion, there is a high chance that sample within the vicinity will exhibit similar, and hopefully improved, performance. This section adapts this idea by considering perturbation in the objective function rather than the input variables; a sample with high performance according to a lower fidelity (LF) simulator is likely to have a similar level of performance according to a higher fidelity (HF) simulator. As a result, utilising lower fidelity simulators can infer suitable areas to sample higher fidelity points, but at a lower computational cost. Generalising SuS to consider multiple levels of fidelity has already been done for the purposes of reliability [78]. However, it has not been done for SuS for optimisation, or at all for any robust optimisation method to the author's knowledge.

The pseudocode for the MF-RSS algorithm is provided in algorithm 2. The algorithm begins by first running the RSS algorithm on the LF simulator. The fact that RSS is applied to a LF simulator will be stated with the superscript LF on relevant variables. Once this is complete, the most promising samples are chosen as 'fidelity seeds'. These fidelity seeds and an additional neighbourhood sample for each are then reevaluated on the HF model and re-sorted according to their new objective values. A proportion of the best performing fidelity seeds are taken to a second optimisation stage. Here $N_T^{HF} - N_\epsilon^{HF}$ neighbourhood samples are generated for each of the new seed samples. The neighbourhood of the seed samples are searched to try and locate the local robust Pareto optimal, i.e. there may be a value within $\mathbf{x}_\epsilon$ that has higher effective performance than $\mathbf{x}$. Once the computational budget is exhausted, solutions with the required number of neighbourhood solutions are sorted according to their HF effective values, with the best taken to be solutions of Eq. (3.4).

Algorithm 2 MF-RSS Algorithm

```
1:  Set LF and HF Computational Budgets.
2:  While Budget Required< $E_N^{LF}$
3:      Nominal SuS procedure.
4:      Update $B^{LF}$.
5:  End
6:  Take $X_{Nom}^{LF}$ as initial solutions for Robust stage.
7:  Set $N_T = 1$.
8:  While Budget Required< $E_R^{LF}$
9:      Check $N_\epsilon^{LF}$ for $X_{Seeds}^{LF}$
10:     If $N_\epsilon^{LF} < N_T^{LF}$
11:         Generate $X_T - X_\epsilon$ Neighbourhood Samples
12:     End
13:     Generate $X_{Cand}^{LF}$ and populate conditional level
14:     Update $B^{LF}$
15:     Sort samples into $S_T$ and $S_I$
16:     For $n_s^{LF}$ best samples in $S_I$
17:         Generate $N_T^{LF} - N_\epsilon^{LF}$ Neighbourhood samples
18:     End
19:     Update $S_T$ and select best samples as $X_{Seeds}^{LF}$
20:     Reduce $N$
21:     Set $N_T^{LF} = min(Ld, M)$
22: End
23: Select final $X_{Seeds}^{LF}$ as fidelity seeds.
24: Evaluate fidelity seeds and neighbourhood samples on HF model and sort.
25: Generate $N_T^{HF} - N_\epsilon^{HF}$ neighbours for proportion of best fidelity seeds.
26: Search neighbourhood samples for local robust optimal solutions.
```

3.4 Numerical Experiments

This section showcases the performance of both RSS and MF-RSS via two well-known test problems. The latter is then applied to an industrial CFD case study.

3.4.1 RSS Illustrative Example

The ZDT1 problem [79] was chosen to measure the performance of RSS. The problem involved attempting to minimise two competing objective functions whilst accounting for input uncertainty:

$$\begin{aligned} f_1 &= x_1, \\ f_2 &= \left(1 + 9\sum_{i=2}^{d} \frac{x_i}{d-1}\right)\left(1 - \sqrt{\frac{x_1}{(1 + 9\sum_{i=2}^{d} \frac{x_i}{d-1})}}\right), \\ \mathbf{x} &\in [0,1], \end{aligned} \tag{3.6}$$

where $d = 5$. To account for input uncertainty, the neighbourhood of $\mathbf{x}$ was defined as

$$\begin{aligned} \mathbf{x}_\epsilon =& \mathbf{x} + \boldsymbol{\epsilon}, \\ \epsilon_i \sim& U(-0.025, 0.025), \\ i =& 1, ..., d, \end{aligned} \tag{3.7}$$

and the effective values (Eq. 3.3) used for the optimisation procedure. This allowed input values to vary by up to 5% with respect to each input dimension. The RSS algorithm was utilised with a total computational budget of 5000 evaluations, of which 2000 were used in the initial optimisation stage, and the remaining 3000 in the robust optimisation stage. The Pareto front according to the RSS algorithm is presented in Figure 3.1, alongside the true robust Pareto front and the true nominal Pareto front for comparison.

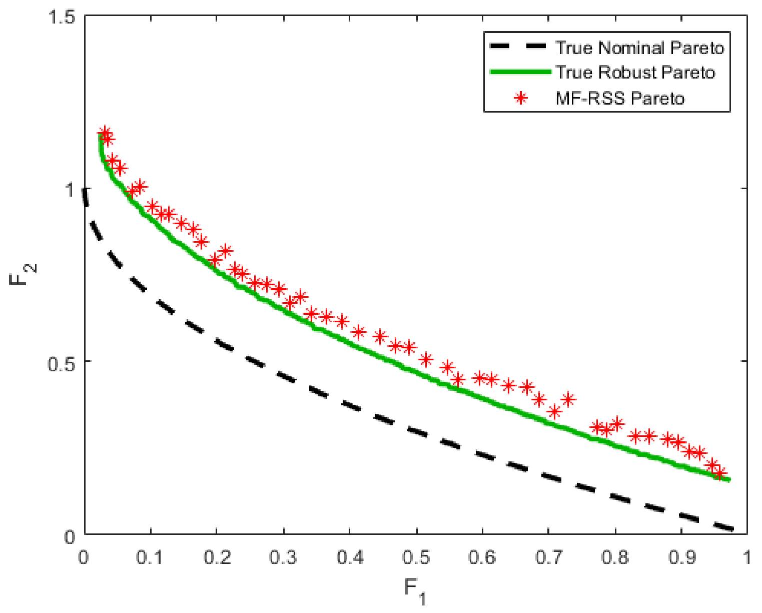


Figure 3.1: **ZDT1 Pareto Fronts:** The dashed black line makes up the nominal Pareto front, the solid green line the true robust Pareto front and the red stars the RSS Pareto front.

The true robust Pareto front (green line) was obtained by utilising the approach described in [70], with $5d$ neighbours evaluated for each sample, and unlimited computational budget. In general, the RSS algorithm produced a Pareto front (red stars) exhibiting similar performance to that true robust Pareto front, but with far less computational expense. As expected, the nominal Pareto front (dashed black line) dominates both of the other Pareto fronts. However, perturbing the input variables of the nominal Pareto front actually results in infeasible solutions; input values that lie outside of bounds of the problem. As such, the robust solutions would both dominate the nominal solutions in the face of input uncertainty. A selection of other benchmark problems described in [71] where also used to test the RSS algorithm, but not presented here to avoid repetition of similar results.

3.4.2 MF-RSS Illustrative Example

The ZDT2 problem was adapted to become a multi-fidelity problem [80] in order to test the MF-RSS algorithm. The original ZDT2 functions were designated as the high-fidelity (HF) functions of the problem:

$$\begin{aligned} f_1 &= x_1, \\ f_2 &= \left(1 + 9\sum_{i=2}^{d} \frac{x_i}{d-1}\right)\left(1 - \left(\frac{x_1}{1 + 9\sum_{i=2}^{d} \frac{x_i}{d-1}}\right)^2\right), \\ \mathbf{x} &\in [0,1], \end{aligned} \tag{3.8}$$

where $d = 5$. The low-fidelity (LF) functions were designated to be the Taylor expansion of the HF functions. To account for input uncertainty, the neighbourhood of $\mathbf{x}$ was defined as in Eq. (3.7) and the effective values used for the optimisation procedure. The overall computational budget for the problem was chosen to be 500 HF evaluations. The comparative costs between the respective fidelities was chosen to be 1 HF evaluation $\approx$ 20 LF evaluations, to match that of the industrial test case. As a result, the budget for the MF-RSS algorithm was 5000 LF evaluations, with 2000 and 3000 in the first and second stages respectively, and 250 HF evaluations. The output from the RSS stage of the algorithm is showcased in figure 3.2. The blue dots show the objective values for every LF evaluation, and are contained in the bank of solutions. The effects of the nominal optimisation stage can be seen in the increased density of samples in more promising areas. There is a noticeable sparsity in the top right of the figure, i.e. an area with poor performance, and increased density toward the Pareto front. The fidelity seeds (red stars) are located in close proximity to the samples with the best performance. The samples exhibiting 'superior' performance either possess lower effective objective performance, or lack the required number of neighbourhood solutions to be considered as a fidelity seed.

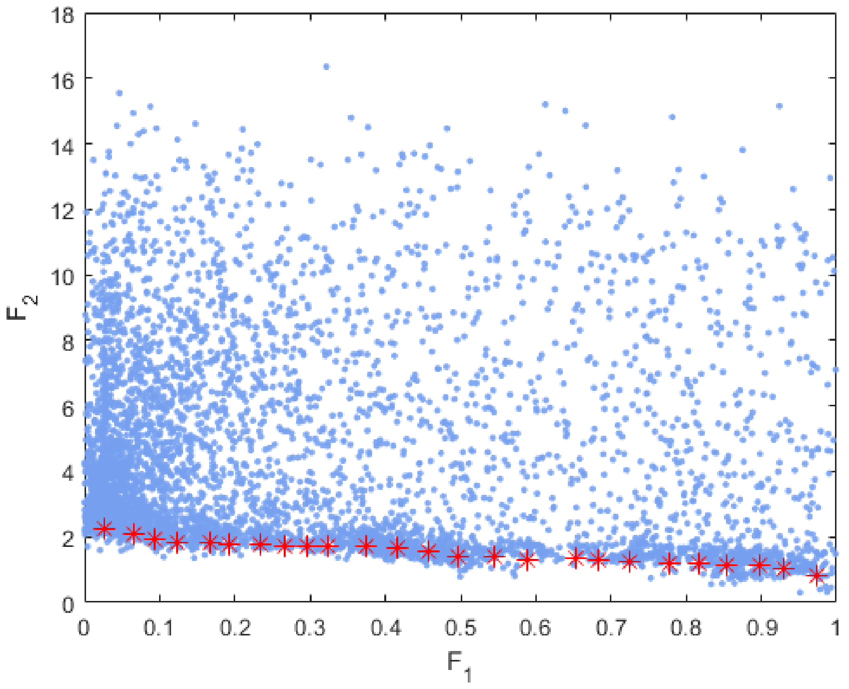


Figure 3.2: **ZDT2 LF Output Space:** The blue dots represent the bank of LF solutions, while the red stars the fidelity seeds.

The fidelity seeds were then evaluated on the HF simulator, along with approximately $2d$ neighbourhood samples for each. Figure 3.3 depicts the output of these evaluations, alongside the true robust Pareto front (green line) and the nominal Pareto front (dashed black line). The fidelity seeds (red stars) lie within close proximity to the robust Pareto front, and crucially, their neighbourhood samples (blue dots) also exhibit similar performance. Moreover, these solutions were obtained with a far more stringent computational budget. Once again the nominal Pareto front dominates all other solutions, but would again be infeasible in the face of input uncertainty. Unfortunately there are no recognised suites of test problems for multi-fidelity robust optimisation problems within the literature. To address this issue, a selection of other SF benchmark problems described in [71] where also extended in a similar format as the illustrative example presented in this chapter. These problems were used to test the MF-RSS algorithm, but not presented here to avoid repetition of similar results.

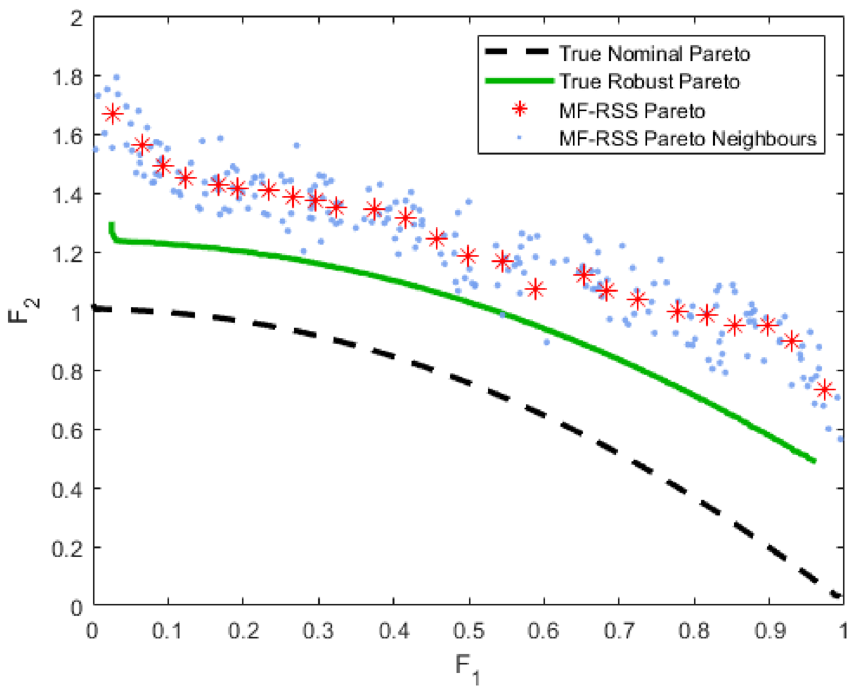


Figure 3.3: **ZDT2 HF Pareto Fronts:** The dashed black line makes up the nominal Pareto front and the solid green line the true robust Pareto front. The HF output of the fidelity seeds is shown by the red stars, and their respective neighbourhood solutions by the blue dots.

3.4.3 Aerofoil Case Study

The use of Computational Fluid Dynamics (CFD) [15] models are widespread in engineering design. Such models are often computationally expensive, and adaptable to multiple levels of fidelity for a given problem. The goal of the aerofoil test case was to obtain aerofoils of a turbine blade that offered maximal lift-to-drag (L/D) ratio and minimal maximum blade thickness subject to perturbations in the input variables caused by uncertainty. Each aerofoil was produced according using the Class-Shape Transformation (CST) method [81] and twenty weighting coefficients (Table 3.1) that define the aerofoil thickness at various locations along the surface. The problem consisted of two levels of fidelity. The LF model involved solving the aerofoil over a range of angles of attack (0-10 degrees) using XFOIL software, which performed a potential flow calculation without taking into account viscosity or a boundary layer. The HF model involved solving the aerofoil via k-ω RANS using ANSYS software. The overall computational budget for the problem was chosen to be 750 HF evaluations. The comparative costs between the respective fidelities was 1 HF evaluation $\approx$ 20 LF evaluations. As a result, the budget for the MF-RSS algorithm was 10000 LF evaluations, with 5000 and 5000 in the first and second stages respectively, and 250 HF evaluations. The MF-RSS algorithm was run with maximum neighbourhood thresholds of $5d$ and $1d$ for the LF and HF stages respectively, and input uncertainty defined as in Eq. (3.7).

Aerofoil Input Variables		
Input Variable	Lower Bound	Upper Bound
Au1	0.1	0.3
Au2	0.1	0.5
Au3	0.0	0.6
Au4	0.0	0.7
Au5	-0.1	0.7
Au6	-0.1	0.7
Au7	-0.1	0.7
Au8	0.0	0.6
Au9	0.0	0.5
Au10	0.1	0.4
Al1	0.1	0.3
Al2	0.1	0.4
Al3	0.1	0.5
Al4	0.0	0.6
Al5	0.0	0.5
Al6	0.0	0.4
Al7	-0.1	0.4
Al8	-0.2	0.4
Al9	-0.2	0.3
Al10	-0.2	0.3

Table 3.1: Input parameter ranges for aerofoil case study.

The output from the RSS stage of the algorithm is showcased in figure 3.4. The blue dots again depict the bank of LF solutions. As in the previous example, the nominal optimisation stage can be seen in the increased density of samples in more promising areas, ensuring a more efficient use of the computational budget. The fidelity seeds (red stars) form a robust Pareto front according to the LF model. This is highlighted by a high level of performance whilst omitting solutions with either ‘superior’, but less robust performance, or those that lack $5d$ neighbourhood samples.

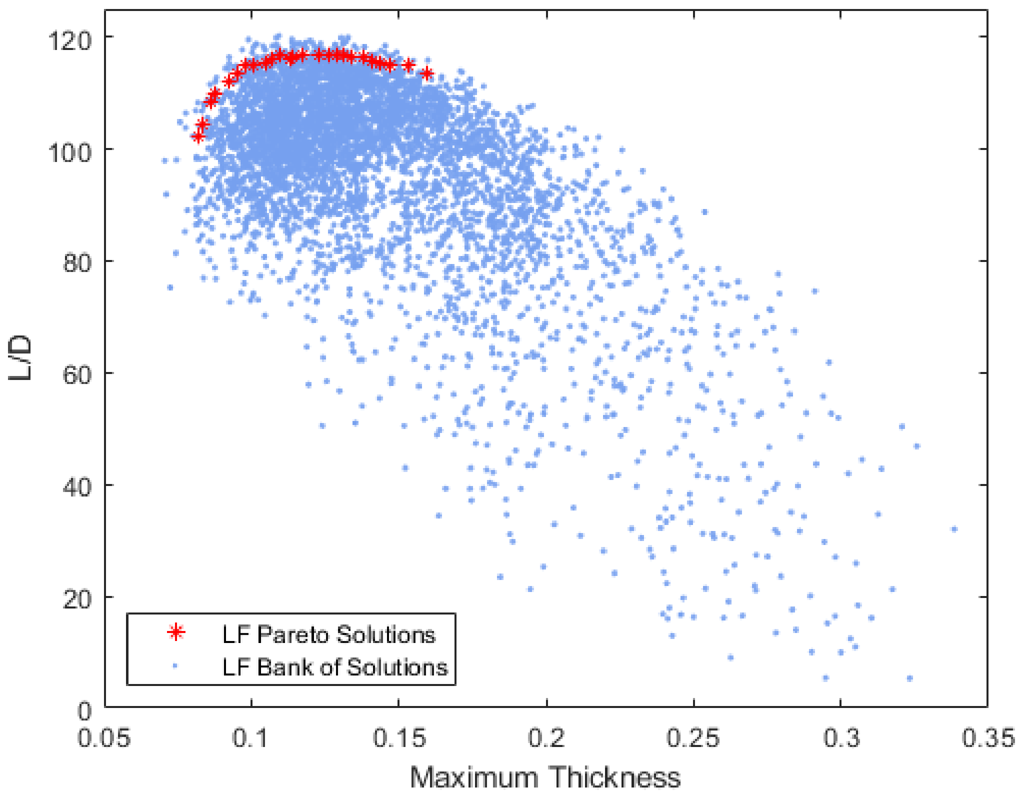


Figure 3.4: **LF Output Space:** The blue dots represent the output of the LF samples evaluated in XFOIL. The red stars represent the fidelity seeds.

Figure 3.5 illustrates the MF stage of the MF-RSS algorithm. The fidelity seeds (red stars) and a single neighbourhood sample for each were evaluated on the HF model. Ten of the best performing fidelity seeds were then supplemented with further neighbourhood samples (blue dots) to ensure they reached the required threshold. These were then taken as solutions to the overall aerofoil test case. On examination, there is a reasonable discrepancy between the LF L/D output and HF L/D output of all fidelity seeds. This is somewhat expected, as the LF model makes more assumptions than the HF model, which can often result in overestimating this particular objective. There is more discrepancy between some solutions than others however, highlighting that high performance according to the LF model does not guarantee high performance according to the HF model. Moreover, the robustness of the two solutions with the lowest maximum thickness was also reduced, with a noticeable drop off in the performance of their neighbourhood samples. Again, this could be due to the overly-simplistic assumptions of the LF model in these areas. Nevertheless, the best performing fidelity seeds that were chosen to proceed with all exhibited strong performance and maintained their robustness from the previous stage, with all neighbours also possessing similar performance.

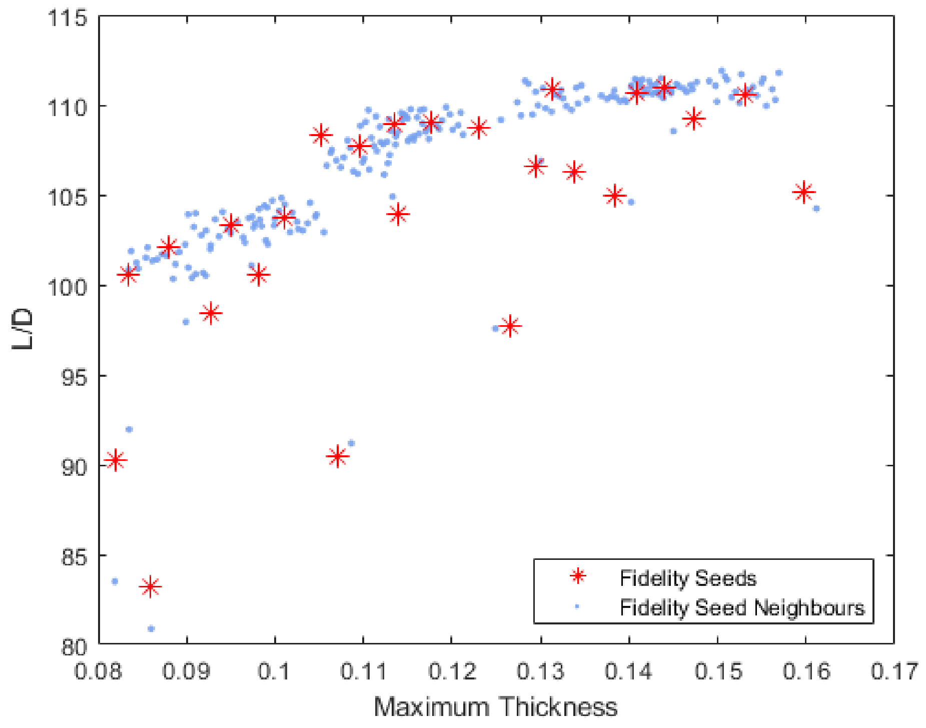


Figure 3.5: **LF Output Space:** The blue dots represent the output of the LF samples evaluated in XFOIL. The red stars represent the fidelity seeds.

3.5 Concluding Remarks of Chapter

This chapter presented a novel method that extends the subset simulation algorithm for the purposes of robust optimisation of computationally expensive models, denoted MF-RSS. This directly addresses the main goal of the research in this thesis, and was motivated by the need to address some of the problems experienced by existing methods. To this end, the method employs several strategies to maximise efficiency and reduce the computational costs involved. Utilising an initial nominal optimisation stage ensures more computational resource is saved for regions of higher performance. The bank of solutions provides significant computational savings by reducing the number of unnecessary evaluations of neighbourhood samples. Further, adapting the number of samples in a conditional level and the neighbourhood threshold ensure that the greatest computational expense is reserved for samples with the most promising robust performance. Finally, generalising the method to incorporate multiple levels of fidelity further reduces computational cost and increases the applicability of the method to industrial-scale problems. Results showcased the ability of MF-RSS to locate designs with high performance and low variability at a reasonable computational cost. Future work includes applying the method to problems with more than two levels of fidelity. However, in particularly computationally constrained problems, the MF-RSS method may be inapplicable, even with the measures to boost computational efficiency. This potential limitation provided the motivation for the next two chapters, which present a surrogate modelling solution to this issue which facilitates the application of the RSS method to a wider array of problems.

Chapter 4

Gaussian Process Emulation

This chapter presents a thorough overview of Gaussian process emulation. The chapter begins by introducing the notion of surrogate modelling, describing the benefits of utilising such methods when dealing with computationally expensive models. In particular, Gaussian process emulation is highlighted as a suitable surrogate method, due to its statistical nature providing a measure of uncertainty associated with its own predictions. Next, the mathematical background of Gaussian processes are discussed, which provide the foundation for the Gaussian process emulation. A step-by-step walk-through of the construction of a Gaussian process emulator (GPE) is then presented, including guidance on necessary implementation details and validation methods. A suitable test problem is then presented to demonstrate the method, and provide a visual display of the preceding theoretical foundations. Finally, the chapter ends by discussing several enhancements of Gaussian process emulation, namely utilising adaptive sampling schemes, factoring in uncertainty within the input variables and exploiting multi-fidelity training data. These enhancements, and the rest of the chapter, provide the foundation for the robust optimisation framework discussed in chapter 5.

4.1 Surrogate Modelling

Computational models play a pivotal role in practically all fields of science and engineering to simulate the behaviour of real-world systems [39]. Such models are designed to take a set of input variables, which contain particular information regarding the physical system under study, such as material properties, in order to produce a number of output variables that would occur in the event of actually performing the physical experiment. As a result, computational models can be framed in a mathematical context of providing a direct mapping from a d-dimensional input space to a p-dimensional output space, $f(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^p$. However, for most computational models, the complexity involved in modelling the underlying physical system results in a lack of closed form expression. As a result, the exact nature of this mapping f is not explicitly known and is generally referred to as 'black-box'; any output for a specific set of input values is unknown until the model is evaluated. Moreover, it is common for such models to be deterministic by design, meaning that evaluating the model multiple times at the same set of input variables always produces the same output.

Another common implication of the complexity of the physical system is a high computational cost involved in evaluating the computational model. The exact computational cost is dependent on a variety of factors, from hardware capability to the actual specification of the model itself. However, for the purpose of this thesis, a computational model is considered expensive in the case that its associated computational cost prevents any type of analysis which requires a relatively high number of model evaluations. The type of analysis which holds particular interest in this thesis is robust optimisation. One of the motivations behind the MF-RSS algorithm discussed in chapter 3 was performing this analysis on expensive models, whilst complying with the computational costs involved. However, in certain cases, even this approach may require too many evaluations of the computational model. A popular approach to combat extreme computational constraints is to utilise a surrogate model to act in place of the expensive computational model. Any analysis can then be performed using the surrogate model.

Evaluating a computational model for a given set of input settings is commonly referred to as a computer experiment [82]. Given the expense associated with evaluating the computational model, it may only be feasible to evaluate the model n times, where n is typically relatively small. A surrogate model, also known as a metamodel or emulator and denoted $\eta(\cdot)$, utilises a set of model evaluations $\mathcal{D} = (\{\mathbf{x}_1, \mathbf{y}_1\}, ..., \{\mathbf{x}_n, \mathbf{y}_n\})$ in an attempt to infer the behaviour of the model at any given input configuration:

$$\mathbf{y} = f(\mathbf{x}) \approx \eta(\mathbf{x}). \tag{4.1}$$

In this sense, a surrogate model can therefore be thought of as an inexpensive approximation to a more computationally expensive model. The set of model evaluations $\mathcal{D}$ is referred to as the training set, and the output of the surrogate model is heavily dependent on the information provided by these values. Consequently, selecting the input configurations that provide the most information regarding the behaviour of the underlying computational model is paramount to the success of a surrogate model. This procedure is commonly referred to as the design of experiments (DoE) [83]. Wherever possible, the DoE should be guided by expert opinion regarding which areas are of high interest to sample from for the training set. However, when this is not possible, a more generalised approach is needed. In particular, in order to fully approximate the computational model throughout its input domain, it is usually important to ensure that any training data is distributed evenly across this space. This task has provided the motivation behind a number of space-filling methods, with the most popular discussed in detail in [1, 84]. A common choice, and the one employed throughout this research for DoE purposes, is to employ maximin, stratified Latin Hypercube sampling (LHS) [85] as it provides a more evenly distributed sample space than uniform sampling, and is much computationally cheaper than Monte Carlo sampling methods. In particular, a candidate point is chosen such that is minimises the maximum distance between other points, hence the maxmin term, whilst ensuring that points are evenly distributed across each dimension.

However, before being able to initiate the DoE, it is necessary to know what value is assigned to n, that is how large is the training set. By the motivation of the problem, this value will likely possess a relatively low upper limit, however in order to possess some confidence in the ability of the surrogate model, thought needs to be given to its lower limit. This value is often dependent on a variety of factors, such as complexity of the computational model and the various assumptions of the surrogate model itself, however a generally accepted rule of thumb is to begin with the basis of at least $n = 10d$ training samples, where d is the number of input variables of the computational model [86].

Generally, an effective surrogate model should satisfy the following properties [87]:

- Accuracy - The output of the surrogate model should provide a good approximation to the output of the underlying computational model for all potential input configurations.
- Speed - The surrogate model should provide a fast approximation to the original computational model, at a fraction of the computational cost.
- Uncertainty - A surrogate model should possess the means to provide a realistic expression of uncertainty at any input configuration within the input domain. In particular, when evaluated at known input values contained in $\mathcal{D}$, the surrogate model should reproduce the corresponding known outputs with little-to-no error.

There is extensive literature regarding the use of surrogate models in a vast number of areas, for example climate modelling [89], environmental science [90], medicine [91], quality control [92], to name a few. With such a wide range in areas of application, an extensive number of different surrogate modelling methods have emerged over the years. Some of the main methods include response surface methods (RSM) [93], Taylor expansions [36], support vector machines (SVM) [38], neural networks (NN) [35], radial basis functions [37] and polynomial chaos expansions [94]. These methods each possess various advantages and disadvantages as summarised in [1]. However, a common issue is that none of these methodologies are statistical in nature, meaning that they each require supplementary tools in order to provide an expression of the uncertainty introduced when employing a surrogate model. An alternative method which does not have this issue is the Gaussian process emulator (GPE). The GPE is defined by a mean function which provides an inexpensive approximation to the computational model, and a covariance function which provides a measure of output uncertainty at each set of inputs [39]. The GPE and its enhancements are the subject of the rest of this chapter.

4.2 Gaussian Process Emulation

Gaussian process emulation is a surrogate modelling method that originated in the area of geostatistics [95, 96], and has been a popular choice for emulating computational models since it was first done in 1989 by Sacks et al [97]. The basis of the method involves treating the unknown output of a computational model as a realisation of type of stochastic process known as a Gaussian process, which provides a statistical approximation of the output. The reasoning behind utilising a Gaussian process in particular, is due to the fact that it possesses a number of desirable properties, such as analytical tractability and flexibility, as well as the fact that is often realistic. A major benefit of the statistical nature of a GPE is the fact that the method not only provides an approximation of the underlying model, but is also able to quantify the uncertainty that arises from the fact that the output of the computational model has not been observed at all input values.

A significant decision when employing a GPE is whether to construct from either a classical or a Bayesian perspective. The classical, or frequentist, approach is commonly referred to as Kriging [98], and involves constructing the surrogate to interpolate the training data and estimate unknown points using a linear combinations of known observations. Moreover, model parameters are considered unknown, but fixed quantities that can be estimated from the data. On the other hand, the goal of the Bayesian approach is to account for all sources of uncertainty, such that model parameters are assigned probability distributions as oppose to point estimates. These probability distributions are constructed according to any beliefs the decision-maker may possess regarding the nature of the model parameter in question, and adjusted accordingly in the light of any additional relevant information. Indeed, updating prior beliefs and associated probability distributions is done using a simple mathematical formulation known as Bayes' theorem:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}.$$

Here, θ represents some quantity of interest, such as a model parameter, and $p(\theta)$ its probability distribution prior to observing several realisations of the model, contained in $\mathcal{D}$. This probability distribution is unsurprisingly referred to as the prior within Bayesian literature. The denominator $p(\mathcal{D})$ represents the evidence of the data, whilst $p(\mathcal{D}|\theta)$ represents the likelihood of observing the values contained in $\mathcal{D}$, depending on the nature of θ. The product of the prior and likelihood produces the posterior distribution, $p(\theta|\mathcal{D})$, which contains the updated beliefs regarding the nature of θ in light of the new information provided in $\mathcal{D}$. There is no consensus over which of these perspectives is superior, but this thesis adopts the Bayesian approach described in [99] due to the attraction of being able to attempt to quantify all forms of uncertainty that can arise during the problem formulation.

4.2.1 Gaussian processes

The normal, or Gaussian, distribution is an exceptionally important probability distribution as it accurately portrays many natural phenomena. Moreover, the central limit theorem states that the distribution of sample means approximates a normal distribution as the sample size approaches infinity, regardless of the shape of the population distribution. As a result, the normal distribution is extremely popular in a vast array of fields. Given that X is a normally distributed random variable, that is $\mathcal{N}(\mu, \sigma^2)$, its probability density function (PDF), denoted $p(x)$, is defined

$$p(x) = \frac{1}{\sqrt{2\pi^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right], \tag{4.2}$$

where μ and σ represent the mean and variance of the distribution. Equation 4.2 describes the PDF for the univariate case which is defined over scalars, i.e. $X \in \mathbb{R}$. This can be generalised for the multivariate case where given a random vector $X \in \mathbb{R}^d$ follows a multivariate Gaussian distribution, its PDF is defined

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu) \right], \tag{4.3}$$

where $\Sigma = \sigma^2 \mathbf{C}$ denotes the covariance matrix, which is the product of the variance and correlation matrix. The mean, μ, is defined in the same vein as the univariate case, but in this case as a vector, $\mu \in \mathbb{R}^d$. The operators $|\cdot|$ and $(\cdot)^T$ represent the determinant the transpose of a matrix respectively.

The Gaussian distribution enjoys several beneficial properties which are stated below without proof. Formal proofs are considered beyond the scope of this thesis and provided in [100]. Given $X \sim \mathcal{N}(\mu, \Sigma)$ with PDF given by Eq. 4.3, and partitioning $\mathbf{x}$, μ and Σ as follows

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

the following properties hold true:

1. The marginal distributions are Gaussian

$$\begin{aligned}\mathbf{x}_1 &\sim \mathcal{N}(\mu_1, \Sigma_{11})\\ \mathbf{x}_2 &\sim \mathcal{N}(\mu_2, \Sigma_{22})\end{aligned}$$

2. The conditional distributions are Gaussian:

$$\begin{aligned}\mathbf{x}_1|\mathbf{x}_2 &\sim \mathcal{N}(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})\\ \mathbf{x}_2|\mathbf{x}_1 &\sim \mathcal{N}(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{x}_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})\end{aligned}$$

3. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are independent, then any linear combination of the two is Gaussian:

$$\begin{bmatrix} a & b \end{bmatrix}\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} a & b \end{bmatrix}\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} a & b \end{bmatrix}\begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}\right)$$

Figure 4.1a displays a bivariate Gaussian distribution, given by the blue lines. Additionally, the red line represents the situation where the value for x_2 is known exactly. Figure 4.1b presents two univariate Gaussian distributions based off the information provided in figure 4.1a. The distribution given by the blue line represents the distribution found by marginalising the bivariate distribution. The distribution given by the red line represents the conditional distribution $p(x_1|x_2)$, using the exact value for x_2. The variation in the shape of the two distributions, such as the increased height and reduced width of the red line, highlights the effect of the conditional information.

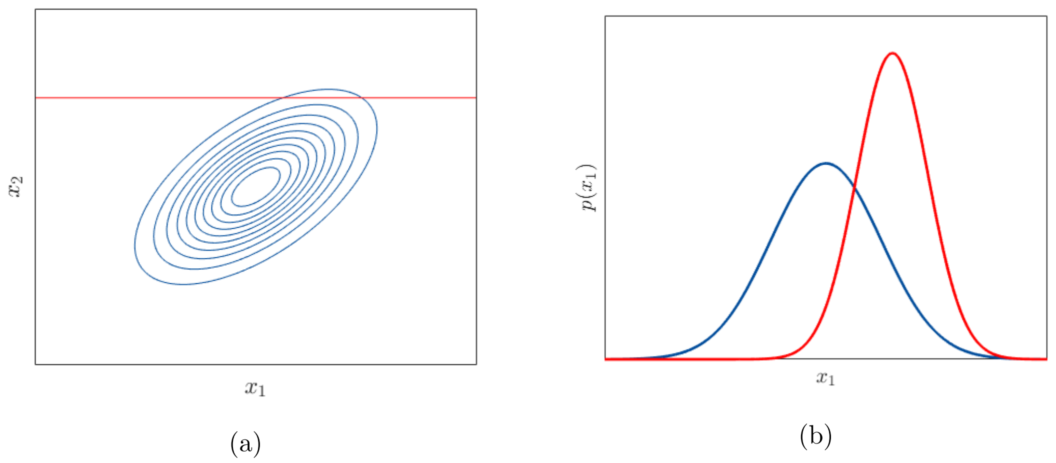


Figure 4.1: **Gaussian Distributions:** (a) depicts a bivariate Gaussian distribution (b) provides the marginalised (blue) and conditional (red) univariate Gaussian distributions.

A Gaussian process is a generalisation of the Gaussian distribution which provides a distribution over functions, i.e. the random variable is a function rather than a scalar or vector [88]. In particular, given some input $\mathbf{x}$, the Gaussian process outputs a specification for a Gaussian probability distribution rather than just a point-estimate. Thus, for any given set of inputs, their output values are distributed joint-normally. Crucially, this applies regardless of whether the combination of inputs correspond to observed points, i.e. points in the training set, or unobserved points. As a result, the latter can be conditioned on observed values from the training set, which is what ultimately drives the GPE method.

4.2.2 Constructing the Gaussian Process Emulator

To begin, consider a computationally expensive model $f(\cdot)$, such that evaluating the model at a set of inputs $\mathbf{x}$ provides an exact output $f(\mathbf{x})$. Prior to actually evaluating the model, the output value at $\mathbf{x}$ is unknown, and therefore under the Bayesian paradigm, can be treated as a random variable and assigned a probability distribution based off prior beliefs about its behaviour. In particular, the GPE is based under the assumption that the output of the computational model is modelled by a Gaussian process. To construct the GPE, the model output is first assigned a Gaussian process prior:

$$\eta(\mathbf{x})|\beta, \sigma^2, \psi \sim \mathcal{GP}\big(M(\mathbf{x};\beta), V(\mathbf{x},\mathbf{x}';\psi)\big). \tag{4.4}$$

Here the mean function, $M(\mathbf{x})$, is designed to capture the global trend of the model output, and is defined as

$$M(\mathbf{x};\beta) = \mathbf{h}(\mathbf{x})^T\beta, \tag{4.5}$$

where $\mathbf{h}(\cdot) \in \mathbb{R}^q$ is a vector of a user-defined real-valued function that maps $\mathbb{R}^d \to \mathbb{R}^q$ and β is a vector of q unknown coefficients. In comparison to the mean function, the covariance term controls the local behaviour and reflects any prior assumptions regarding the smoothness of the model output. It is defined by

$$V(\mathbf{x},\mathbf{x}';\psi) = \sigma^2 c(\mathbf{x},\mathbf{x}';\psi), \tag{4.6}$$

where σ^2 represents the variance of the model, and ψ is the smoothness parameter which controls the behaviour of the correlation function, $c(\mathbf{x},\mathbf{x}';\psi)$. Several realisations of a zero mean Gaussian process are presented in figure 4.2 to illustrate how a GP prior may behave.

The motivation behind placing a Gaussian process prior on the model output is to exploit the properties described in section 4.2.1. Specifically, given that any finite set of points taken from a Gaussian process possess a multivariate Gaussian distribution, according to the second property from 4.2.1. Consequently, the model output at any unobserved input value $\mathbf{x}^*$ can be conditioned on already observed values, $\mathcal{D}$, and follows a normal distribution:

$$\eta(\mathbf{x}^*)|\mathcal{D}, \sim \mathcal{N}(M(\mathbf{x}^*), V(\mathbf{x}^*,\mathbf{x}^*)) \tag{4.7}$$

with some mean and covariance functions $M(\mathbf{x}^*)$ and $V(\mathbf{x}^*,\mathbf{x}^*)$ respectively. The set of observed model realisations $\mathcal{D} = (\{\mathbf{x}_1, \mathbf{y}_1 = f(\mathbf{x}_1)\}, ..., \{\mathbf{x}_n, \mathbf{y}_n = f(\mathbf{x}_n)\})$ is commonly referred to as the training set for the GPE. By definition, as η follows a Gaussian distribution, the observed model outputs $\mathbf{y}$ also follow a Gaussian distribution which can be defined as

$$\mathbf{y}|\beta, \sigma^2, \psi \sim \mathcal{N}(\mathbf{H}\beta, \sigma^2\mathbf{C}) \tag{4.8}$$

where $\mathbf{H} = (\mathbf{h}(\mathbf{x}_1)^T, ..., \mathbf{h}(\mathbf{x}_n)^T)^T$ contains the evaluations of the GP regression functions

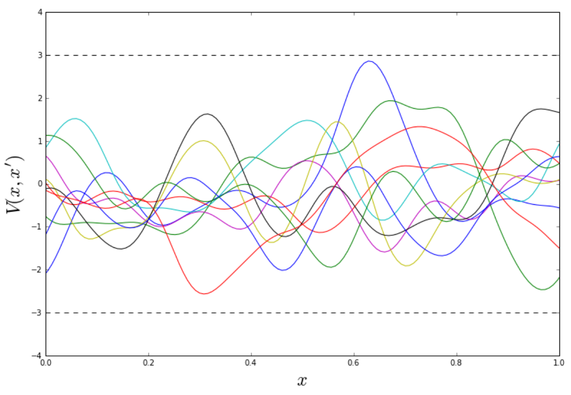


Figure 4.2: Random draws from a zero mean Gaussian process; the dashed lines represent $\pm 3\sigma$ bounds that contain approximately 99.7% of the realisations.

at the training data input values, and $\mathbf{C}$ is an $n \times n$ matrix with elements given by:

$$C_{i,j} = c(\mathbf{x}_i, \mathbf{x}_j; \psi). \tag{4.9}$$

That is, a matrix containing the correlation between all combinations of the training data inputs. The collection of parameters β, σ^2, ψ that control the behaviour of the GPE are collectively referred to as its hyperparameters, and are generally unknown a priori. As a result, in order to be able to utilise the GPE effectively, the values for these hyperparameters need to be estimated from the training data. Fortunately, the regression coefficients β and the GPE variance, σ^2, can be estimated analytically. The smoothness parameter, ψ, is estimated through maximising an associated likelihood function, which is discussed in further detail in section 4.3.1. Combining Eq. 4.3 and 4.8, the likelihood of β and σ^2 can be expressed as

$$p(\mathbf{y}|\beta, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}} |\mathbf{C}|^{\frac{1}{2}}} \exp\left[-\frac{(y - \mathbf{H}\beta)^T \mathbf{C}^{-1} (y - \mathbf{H}\beta)}{2\sigma^2} \right]. \tag{4.10}$$

Next, a prior is set for the two hyperparameters, with the most popular choice in literature to assign a weak prior for the joint distribution:

$$p(\beta, \sigma^2) \propto \frac{1}{\sigma^2}, \tag{4.11}$$

as this enables the training data to play a larger role in the predictions from the emulator.

Once the prior distribution is assigned, the posterior distribution can be obtained using Bayes' rule:

$$p(\beta, \sigma^2|y) \propto p(y|\beta, \sigma^2)p(\beta, \sigma^2). \tag{4.12}$$

Next, reparameterising Eq. 4.10 yields a normal posterior distribution for β, complete with mean estimate $\hat{\beta}$

$$\hat{\beta} = (\mathbf{H}^T\mathbf{C}^{-1}\mathbf{H}^T)^{-1}\mathbf{H}^T\mathbf{C}^{-1}\mathbf{y} \tag{4.13}$$

Combining Eq. 4.10 and Eq. 4.11 whilst separating the result from the distribution of β and performing some slight manipulation gives a inverse-gamma posterior distribution for σ^2 with an unbiased estimator:

$$\hat{\sigma}^2 = \frac{\mathbf{C}^{-1} - \mathbf{C}^{-1}\mathbf{H}(\mathbf{H}^T\mathbf{C}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{C}^{-1})\mathbf{y}}{n-q-2} \tag{4.14}$$

Combining Eq. 4.8, Eq. 4.13 and Eq. 4.14 produces the posterior distribution for model output, conditional on the training data and hyperparameter estimators as

$$\eta(\mathbf{x}^*)|\mathcal{D}, \hat{\beta}, \hat{\sigma}^2 \sim \mathcal{N}(M^*(\mathbf{x}^*), V^*(\mathbf{x}^*, \mathbf{x}^*)) \tag{4.15}$$

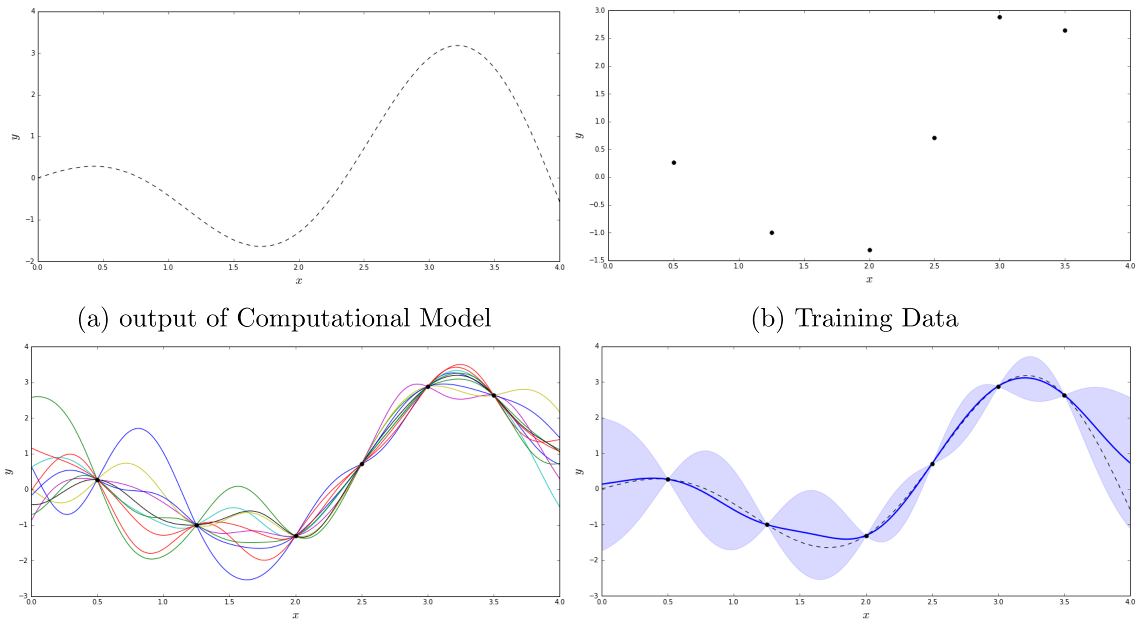
with the posterior mean function given by

$$M^*(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*)^T\hat{\beta} + t(\mathbf{x}^*)^T\mathbf{C}^{-1}(y - \mathbf{H}\hat{\beta}), \tag{4.16}$$

and the posterior covariance function given by

$$\begin{aligned} V^*(\mathbf{x}^*, \mathbf{x}^{*\prime}) =& \hat{\sigma}^2\big[c(\mathbf{x}^*, \mathbf{x}^{*\prime}) - \mathbf{t}(\mathbf{x}^*)^\mathbf{T}\mathbf{C}^{-1}\mathbf{t}(\mathbf{x}^{*\prime}) \\ &+ (\mathbf{h}(\mathbf{x}^*)^\mathbf{T} - \mathbf{t}(\mathbf{x}^*)^\mathbf{T}\mathbf{C}^{-1}\mathbf{H})(\mathbf{H}^\mathbf{T}\mathbf{C}^{-1}\mathbf{H})^{-1} \\ &\times (\mathbf{h}(\mathbf{x}^*)^\mathbf{T} - \mathbf{t}(\mathbf{x}^*)^\mathbf{T}\mathbf{C}^{-1}\mathbf{H})^\mathbf{T}\big], \end{aligned} \tag{4.17}$$

where $\mathbf{t}(\mathbf{x}^*) = (c(\mathbf{x}^*, \mathbf{x}_1; \hat{\psi}), ..., c(\mathbf{x}^*, \mathbf{x}_n; \hat{\psi}))^T$ contains the correlation between $\mathbf{x}^*$ and the training data. Once the GPE is constructed, its mean function can provide an inexpensive approximation to the computational model at all potential input configurations, whilst its covariance function provides a realistic measure of uncertainty regarding the mean approximation. Moreover, when evaluated at a known input, i.e. $\mathbf{x}_i \in \mathcal{D}$, the GPE will produce the associated output value $\mathbf{y}_i$ with zero uncertainty, to reflect the fact that this value is known. Figure 4.3 highlights the effect that conditioning on the training data has



(a) output of Computational Model

(b) Training Data

(c) GP Posterior Distribution Realisations

(d) Final GPE

Figure 4.3: Illustrative example showcasing the concept of Gaussian process emulation.

on the GP priors from figure 4.2. In particular, the dashed black line in figure 4.3a is some function that is to be emulated. The black dots in figure 4.3b represent realisations of the function, which act as training data for the GPE. Figure 4.3c showcases the realisations of the GP posterior; note how they all interpolate at the training points before diverging in between. 4.3d summarises the preceding steps; the dashed line is the true function, the solid blue line is the mean output of the GPE and the shaded light blue regions represent the bounds for 99% of the GP posterior realisations, i.e. the uncertainty bounds for the GPE.

4.3 Implementation Details

The preceding section provided the theoretical background behind employing Gaussian Process Emulation to approximate some computationally expensive function. However, in order to actually implement the GPE, a number of key choices need to be made by the user. The first decision was touched upon at the start of the chapter and is a key stage in all surrogate methods; deciding the number of training points n and which method to employ to sample them. For computationally expensive problems, the first part of this decision is often made for the user, as the computational expense means that there is a stringent upper limit to the number of potential training points available. As a result, it is common to completely exhaust the given computational budget when sampling for training points, as this limit is usually lower than the theoretical optimal number of training points for the actual problem. For the second part of the decision, unless expert opinion dictates differently, the most important criteria for sampling is to ensure that training points are distributed fairly evenly throughout the input space, in order to capture the behaviour of the computational model throughout the entire domain. As mentioned earlier, LHS is a popular choice for this task, and is used throughout this research.

The next decision is deciding on the form of mean function described in Eq. 4.5, and more specifically the regression term $\mathbf{h}(\mathbf{x})^T$. As stated, the goal of the mean function is to capture the global trend of the model output. The regression term is made up of a vector of q user-defined real-valued regression functions, and are defined to represent any prior beliefs on the behaviour of the underlying model. The β term represents a vector of q coefficients that are estimated from the training data (Eq. 4.13), which act as weights for each of the q regression functions. Selecting a mean function that accurately captures the behaviour of the underlying computational model greatly simplifies the task for emulation. However, without expert guidance, and given that such models are often black-box and their output space unknown, selecting such a mean function is often infeasible. Instead, common choices for the regression term include linear regression, $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x})$ or a constant value, such as $\mathbf{h}(\mathbf{x})^T = (1)$ or $\mathbf{h}(\mathbf{x})^T = (0)$. These provide weak assumptions regarding the nature of the model output. and allow the process of emulation to be largely dictated by the covariance term (Eq. 4.6), which itself is heavily influenced by the training data. This leads to the next major choice, determining the make-up of the covariance function.

Similarly to the mean function, the covariance function is the product of a user-defined term, namely the correlation function, or kernel as it is known in some communities, $c(\mathbf{x}, \mathbf{x'}; \psi)$, and a scalar hyperparameter estimated from the data, σ^2. However, unlike the regression term in the mean function, there are certain requirements that a kernel has to pass in order to be eligible for use. Specifically, any covariance matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$, produced by a kernel must be symmetric, i.e. $\mathbf{C} = \mathbf{C}^T$, and positive semi-definite, that is $\alpha^T \mathbf{C} \alpha \geq 0,\ \forall \alpha \in \mathbb{R}^n$. Nevertheless, even with the stricter restrictions, there are still a wide variety of different kernels available. In particular, kernels are often broken down into two categories, stationary and non-stationary. Stationary kernels are functions with output solely determined by the proximity of points within the input space, that is $c(\mathbf{x}_i, \mathbf{x}_j; \psi) = c(|\mathbf{x}_i - \mathbf{x}_j|; \psi)$. On the other hand, non-stationary kernels also rely on the respective position of a point, as well as its proximity to other points. Various kernels from both categories are covered extensively in [101], while several of the most popular are described here. Table 4.1 contains the mathematical formulation for two stationary kernels, the exponentiated quadratic and Matérn family, and one non-stationary kernel, the Brownian. Further, figure 4.4 contains several draws from a zero-mean Gaussian process with each of the three kernels, providing an illustration between the influence the kernel choice has on behaviour of the output.

Kernel Name	Formulation
Exponentiated Quadratic	$\exp\left(-\frac{r^2}{\psi^2}\right), \quad r = \lvert\mathbf{x} - \mathbf{x}'\rvert$
Matérn 5/2	$\left(1 + \frac{\sqrt{5}r}{\psi} + \frac{5r^2}{3\psi^2}\right)\exp\left(\frac{-\sqrt{5}r}{\psi}\right)$
Brownian	$\min(\mathbf{x}_i, \mathbf{x}_j)$

Table 4.1: Common kernel choices

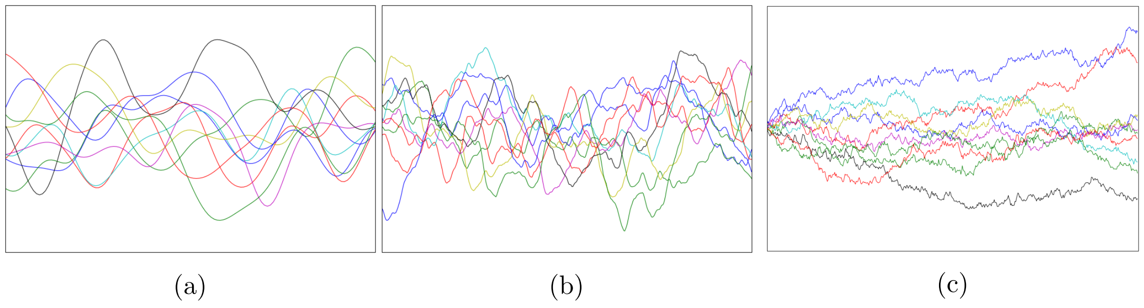
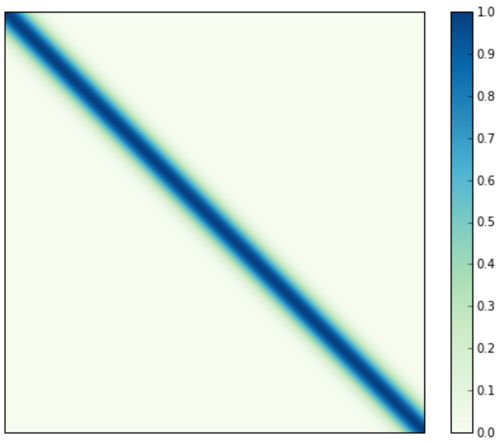
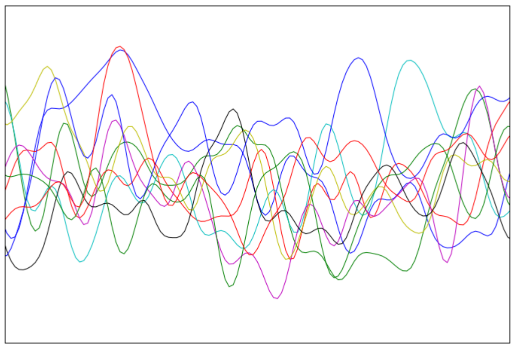


Figure 4.4: **Kernels** Several Gaussian Process draws for different kernels: (a) exponentiated quadratic, (b) Matérn 5/2 and (c) Brownian.

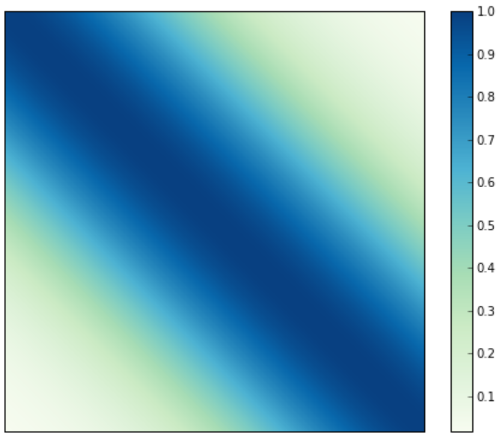
The choice of kernel should reflect the prior beliefs regarding the system under study. The exponentiated quadratic is a popular choice in the literature as it possess attractive mathematical properties, and will now be the sole correlation function utilised throughout the rest of this thesis. However, regardless of the exact choice, a commonality across all kernel functions is the requirement to specify certain hyperparameters. Although some kernels will possess additional hyperparameters, the most common is the hyperparameter that controls the relative smoothness of the kernel, ψ. Given that most computational models possess multiple input dimensions, each with a varying effect on the model output, this smoothness hyperparameter is usually defined as a d-dimensional vector, referred to as automatic relevance determination (ARD) [102], rather than a single value for all input dimensions. That is, for $\psi_i : i \in \{1, ..., d\}$, ψ_i indicates the respective smoothness value corresponding to the i^{th} input dimension. Figure 4.5 demonstrates the effect varying this smoothness parameter can have on the behaviour of the Gaussian Process. Figure 4.5a contains the correlation between the input space for $\psi = 0.05$, with no correlation give by white, gradually scaling up to a correlation of 1 given by dark blue. On inspection, points are only correlated to points within their immediate vicinity. Consequently, the corresponding GP draws displayed in 4.5b show little correlation and highly irregular behaviour. Figure 4.5c contains the correlation for $\psi = 0.5$. This value implies a much stronger correlation even between points relatively far apart, resulting in a much wider blue region. This feeds into the GP draws in 4.5d, which exhibit much smoother behaviour. This is not to say that the former is inferior to the latter; the smoothness values is problem specific, will vary between input dimensions. For problems which naturally possess irregular behaviour and low correlation between points, the former will obviously be more appropriate, and vice versa with problems with more regular behaviour. Indeed, the task of selecting the optimal values for the smoothness hyperparameters is critical due to their influence on the predictive capability of the GPE. The next section details the procedure involved in dealing with this task.



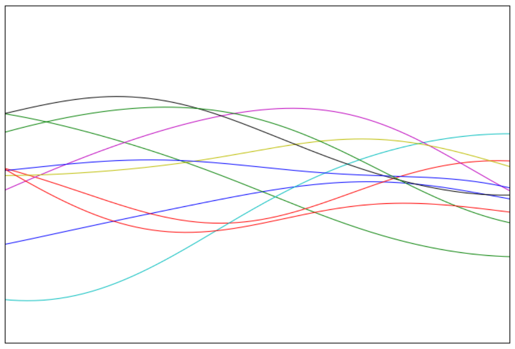
(a) Initial MC Sampling



(b) First Intermediate Failure Region



(c) MCMC Step



(d) Second Intermediate Failure Domain

Figure 4.5: **Effect of Smoothness Hyperparameter** (a) and (c) depict the correlation matrices for $\psi = 0.05$ and $\psi = 0.5$ respectively. Several random draws associated with (a) and (c) are depicted in (b) and (d) respectively.

4.3.1 Estimating the Smoothness Hyperparameters

Outside of evaluating the computational model, the task of estimating suitable smoothness hyperparameter values is often the most computationally expensive portion of constructing the GPE. One option in tackling the problem is to utilise leave-one-out cross-validation

(LOO-CV) to select the values corresponding to the lowest predictive error. Specifically, for a prospective set of hyperparameter values, LOO-CV involves removing a single training point, evaluating the GPE mean output at the missing location based on the remaining training data, and then computing the error between the GPE mean output and the actual value. This is repeated for each of the training points to obtain an overall value for the error for the set of hyperparameter values, known as the LOO-CV value. The method provides a fairly straightforward way to measure the superiority of one set of candidate values against another, with the values corresponding to the lower LOO-CV value deemed the more suitable choice. However, the procedure can be time-consuming, especially as the number of training points increases. Moreover, $\psi_i \in (0, \infty)$, meaning to adequately explore the input space of the hyperparameter values requires a significant number of candidate values to undergo the LOO-CV procedure in order to ensure suitable values are indeed selected. To avoid these issues, a popular strategy is to employ a probability-based approach, such as maximising the posterior mode of the parameter distribution given data can be used, as in [103]. Alternatively, the approach taken in this thesis is to estimate ψ via maximum likelihood estimation (MLE). MLE is a popular method in classical statistics, and involves estimating the hyperparameter values which are most likely to have generated the training data. More specifically, the optimal choice for ψ is set to be the one that maximises Eq. 4.10. Further, to simplify the algebra and improve the stability, it is a common procedure to take the natural logarithm of the likelihood:

$$\ln[f(\mathbf{y}|\beta, \sigma^2)] = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2}\ln|\mathbf{C}| - \frac{(\mathbf{y} - \mathbf{H}\beta)^T\mathbf{C}^{-1}(\mathbf{y} - \mathbf{H}\beta)}{2\sigma^2}. \tag{4.18}$$

After substituting in the analytical definitions for β and σ (Eq. 4.13 and Eq. 4.14) and simplifying the remaining terms, the concentrated log-likelihood is obtained:

$$\ln[f(\mathbf{y}|\beta, \sigma^2)] \approx -\frac{n}{2}\ln(\hat{\sigma}^2) - \frac{1}{2}\ln|\mathbf{C}| \tag{4.19}$$

The concentrated log-likelihood is a close approximation of the full log-likelihood [1] that is often used to increase the efficiency of the task. To maximise Eq. 4.19, the function can either be differentiated with respect to ψ, or treated directly as an optimisation problem. For the latter, a number of methods can be employed, such as the Nelder-Mead algorithm [104], MCMC-based methods [105], simulated annealing [61], genetic algorithms [30], amongst others. However, additional consideration often needs to be taken depending on the nature of the covariance matrix $\mathbf{C}$. By definition, $\mathbf{C}$ is required to be positive semi-definite, however certain scenarios can occur where this is no longer the case and the matrix becomes ill-conditioned. This causes major issues in the estimation of β and σ, where $\mathbf{C}$ has to be inverted, as well as in the evaluation of Eq. 4.19, which contains the determinant of $\mathbf{C}$. The issue of instability can arise due to poorly constructed training data, an overabundance of training data at low input dimensions, and for extreme values of ψ. The presence of any of these issues has a severe impact on the ability to accurately estimate suitable smoothness hyperparameter values. Several methods of addressing this issue have arisen, such as a penalised likelihood function [106] and using a nugget term [107]. The latter is extremely popular, and involves adding a small value to the diagonal of $\mathbf{C}$, known as a nugget, in order to boost its stability, i.e. $\mathbf{C}_\delta = \mathbf{C} + \delta I$. This can address the issue of an otherwise ill-conditioned $\mathbf{C}$, however it also alters the log-likelihood profile and removes the ability of the GPE to interpolate training data exactly. Nevertheless, the change to the log-likelihood is often extremely minor, and for sufficiently small nuggets the effect on interpolation is minimal.

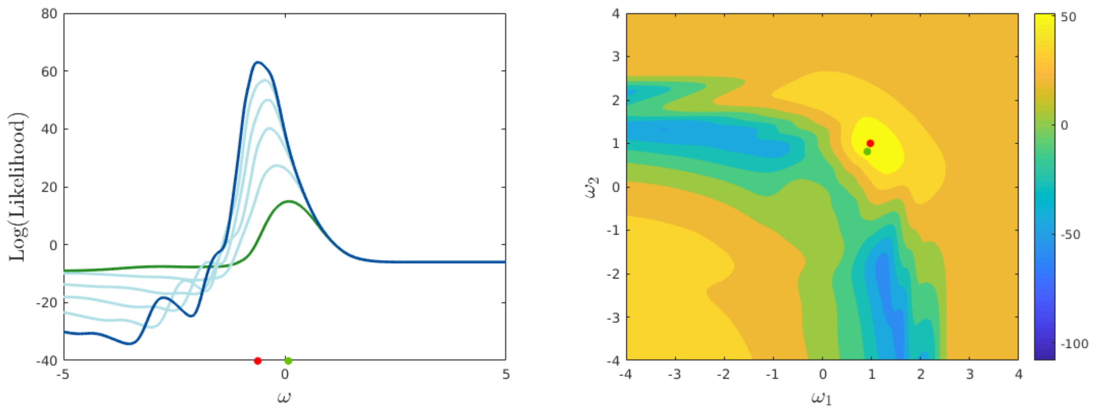
The motivation behind using a nugget is to provide numerical stability when it is needed. However, selecting a suitable value for the nugget is less obvious. In situations where a nugget is not necessary, the presence of the nugget is often detrimental to the performance of the GPE. In the situation where the nugget is necessary but the value is too small, it will not provide enough of an effect to address the ill-conditioned nature of $\mathbf{C}$. Additionally, given that the presence and size of a nugget directly impacts the log-likelihood function, it is often desirable to include the nugget as an extra hyperparameter to optimise simultaneously alongside the smoothness values. One strategy presented in [108] is obtain the theoretical lower bound for the nugget, based on ensuring the condition number of the correlation matrix doesn't not exceed a critical threshold. In particular, it was found that a suitable lower bound could be defined as

$$\delta \gtrapprox \frac{\lambda_n(\kappa(\mathbf{C}) - e^a)}{\kappa(\mathbf{C}()e^a - 1)} = \delta_{lb}, \tag{4.20}$$

where $\kappa(\mathbf{C}) = \frac{\lambda_n}{\lambda_1}$ is the condition number of $\mathbf{C}$, λ_n and λ_1 are the largest and smallest eignevalues of $\mathbf{C}$, and $a \approx 25$. The lower bound for the nugget is used and the modified correlation function fed through to the concentrated log-likelihood in Eq. 4.19. This log-likelihood can then be optimised using one of the standard optimisation methods mentioned above.

Another approach discussed in [109] noted that the presence of a nugget had an overall smoothing effect on the log-likelihood function, but limited impact on the location of optimal values. Consequently, it was shown that optimising the log-likelihood for an overly large nugget value could infer suitable optimal values for more realistic nugget values, or even the case with no nugget present at all. Moreover, smoothing the log-likelihood removes the many local optima often present in the case with small nugget values, allowing a more efficient optimisation process and reducing the likelihood of converging to a local optima rather than the global optima. Specifically, the procedure begins by optimising over a nugget value of $\delta = 10^{-2}$ to obtain a set of optimal smoothness hyperparameter values. The nugget is then decreased by a factor of 10 and the optimisation process restarted using the optimal values from the previous nugget value as a starting point. This is repeated until $\delta = 10^{-12}$ is reached, at which point it is repeated once more for $\delta = 0$. Each of the nugget and smoothness combinations are then evaluated using LOO-CV to obtain the most suitable choice.

Figure 4.6 demonstrates this concept in action. Figure 4.6a displays the log-likelihood for the training data from the function in figure 4.3 for a selection of nugget values. In particular, the green line corresponds to $\delta = 10^{-2}$, and the light blue lines represent the cases from $\delta = 10^{-4}$ to $\delta = 10^{-10}$, and the dark blue line to $\delta = 10^{-12}$. Additionally, the smoothness parameters have been parameterized as $\psi = 10^{\omega}$ to better facilitate plotting. Finally, the green and red dots represent the locations of the optimal values for ω for $\delta = 10^{-2}$ and $\delta = 10^{-12}$ respectively. On inspection, as the nugget value is decreased, so is the overall smoothing effect, but the general behaviour of the log-likelihood remains extremely similar. Indeed the respective optimal locations for a given log-likelihood are in close proximity to preceding optimal locations as the nugget decreases, gradually converging from the location of the green dot to that of the red dot. Figure 4.6b extends the concept to a two-dimensional case, and displays a contour plot of the log-likelihood for the test problem discussed in section 4.4 with $\delta = 10^{-12}$. The green and red dots again represent the optimal ω locations for $\delta = 10^{-2}$ and $\delta = 10^{-12}$ respectively. Once again, it is clear that the two optimal solutions are in close proximity, despite the significant difference in their nugget values.



(a) (b)

Figure 4.6: **Impact of Nugget on Likelihood** (a) Log-likelihood output for a selection of nugget values, as well as optimal ω locations for largest and smallest nuggets. (b) contour plot of log-likelihood output and optimal ω locations.

4.3.2 Diagnostics

The preceding section highlighted how important it is to select suitable hyperparameter values, in order to maximise the performance of the GPE. However, this statement is somewhat subjective without detailing procedures to actually measure the performance of the emulator. Specifically, measuring how well the emulator performs its original goal; approximating the computational model. A series of validation methods and diagnostics are detailed in [110] for this purpose, with several for the most popular summarised in this section. For each, it is common to have a number of model realisations kept separate from the original training set, known as the validation set: $\mathcal{V} = (\{\mathbf{x}_1^V, \mathbf{y}_1^V = f(\mathbf{x}_1^V)\}, ..., \{\mathbf{x}_m^V, \mathbf{y}_m^V = f(\mathbf{x}_m^V)\})$. This validation set is usually obtained via some form of random sampling to ensure there is no bias.

Amongst the most popular of diagnostics is the root mean squared error (RMSE)

$$D^{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\mathbf{y}_i^V - \mathbb{E}[\eta(\mathbf{x}_i^V)|\mathbf{y}^V])^2} \tag{4.21}$$

where m is the number of test points and $\mathbb{E}[\eta(\mathbf{x}_i^*)|\mathbf{y}]$ is the posterior mean function of the GPE defined in Eq. 4.16. The RMSE provides an overall summary of the quality of the approximation of the emulator, with a smaller value indicating superior performance. However, it is susceptible to outliers, such as a small number of large errors, and fails to factor in any of the information regarding the uncertainty at validation points provided by the posterior covariance function. In contrast, the individual prediction errors diagnostic, $D_i^I(\mathbf{y}_i^V)$, makes use of both the posterior mean and covariance:

$$D_i^I(\mathbf{y}_i^V) = \frac{\mathbf{y}_i^V - \mathbb{E}\big[\eta(\mathbf{x}_i^V)|\mathbf{y}\big]}{\sqrt{\mathbb{V}\big[\eta(\mathbf{x}_i^V)|\mathbf{y}\big]}} \tag{4.22}$$

where $\mathbb{V}[\eta(\mathbf{x}_i^V)|\mathbf{y}]$ represents the posterior covariance defined in Eq. 4.25. For a reasonable GPE, the individual prediction errors should have standard student-t distributions conditional on the observations and on the smoothness hyperparameters. As a result, it is required that $D_i^I(\cdot) \leq 3$ as approximately 99% of values lie within 3σ bounds. Any values near or exceeding this bound may indicate inadequate emulator performance in that region of the input space. Although the individual prediction errors is a useful diagnostic, a potential drawback is that it fails to factor in any correlation between of samples in the validation set. A diagnostic that addresses this concern is the Mahalanobis distance between the true output and the GPE mean:

$$D_{MD}(\mathbf{y}^V) = \big(\mathbf{y}^V - \mathbb{E}\big[\eta(\mathbf{x}_i^V)|\mathbf{y}^V\big]\big)^T \big(\mathbb{V}\big[\eta(\mathbf{x}_i^V)|\mathbf{y}\big]\big)^{-1} \big(\mathbf{y}^V - \mathbb{E}\big[\eta(\mathbf{x}_i^V)|\mathbf{y}^V\big]\big), \tag{4.23}$$

where the posterior covariance matrix of the GPE is used to weight the Mahalanobis distance in order to account for correlation amongst predictions. Extreme values (both small and large) of D_{MD} indicate conflict between the emulator and simulator.

However given the computational constraints associated with certain problems, it will not always be possible to save a portion of model evaluations for the sole purpose of validation. In such cases, a common diagnostic is to utilise LOO-CV, which was touched upon earlier in the chapter. LOO-CV is defined as

$$D_{CV} = \frac{1}{M} \sum_{i=1}^{M} \big(\mathbf{y}_i^V - \mathbb{E}\big[\eta(\mathbf{x}_i^V)|\mathbf{y}^{V-i}\big]\big) \tag{4.24}$$

and although it is not as thorough as other diagnostic methods, it can still provide good insight into emulator behaviour.

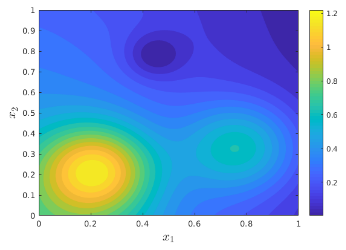
4.4 Illustrative Test Problem

In this section, an illustrative test problem is used to showcase the steps described in section 4. As with SuS-related work, the entirety of the code relating to Gaussian process based methods in this research were written and developed within MATLAB, without the use of any existing toolboxes. Once again, this helped to establish a greater understanding of the underlying fundamentals of the method. This knowledge was essential when factoring in various extensions of Gaussian process emulation that are covered in the last section of the this chapter. To illustrate the GPE method, Franke's function was chosen, with the function defined as

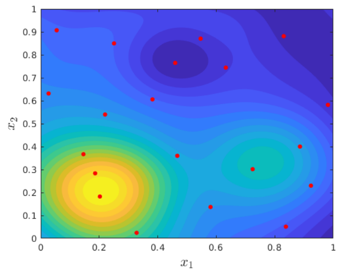
$$\begin{aligned} f(\mathbf{x}) =& \frac{3}{4}\exp\bigg(-\frac{(9x_1-2)^2}{4}-\frac{(9x_2-2)^2}{4}\bigg)+\frac{3}{4}\exp\bigg(-\frac{(9x_1+1)^2}{49}-\frac{(9x_2+1)^2}{10}\bigg) \\ &+\frac{1}{2}\exp\bigg(-\frac{(9x_1-7)^2}{4}-\frac{(9x_2-3)^2}{4}\bigg)-\frac{1}{5}\exp\Big((9x_1-4)^2-(9x_2-7)^2\Big). \end{aligned} \tag{4.25}$$

Franke's function is a commonly used test function for interpolation problems as it exhibits non-linear behaviour similar to what could expected in various engineering problems. Moreover, as in the SuS examples, it possesses two input variables meaning that the process can be easily visualised and more readily understood. Figure 4.7a displays the output space of Franke's function in the form of a contour plot. The function possesses two Gaussian-shaped peaks given by the yellow and lighter blue region, as well as a local minimium given by the dark blue region. Although the function itself is computationally inexpensive to run, in order to best mimic the procedure when utilising a GPE in real-world engineering problems, the training data is limited to $n = 20$ function realisations, in line with the general rule $n = 10d$ mentioned in section 4.1. Figure 4.7b contains the mean output of the GPE in a form of a contour plot, as well as the locations of the training points which are represented by red dots. On inspection, the GPE was able to effectively capture the behaviour of the actual function, including the locations of the local maxima and minima, without any major visible discrepancies. As mentioned several times during this chapter, one of the major benefits of a GPE is the closed-form expression of uncertainty that accompanies each mean prediction. Figure 4.7c demonstrates this feature for the test problem, where the posterior predictive variance is shown via a contour plot. As expected, the uncertainty is lowest in the immediate vicinity of the training points (red dots), and actually reduces to zero at each. Alternatively, the uncertainty is increased in regions that are not in close proximity to any training points, with the highest levels of uncertainty near the boundaries of the input domain.

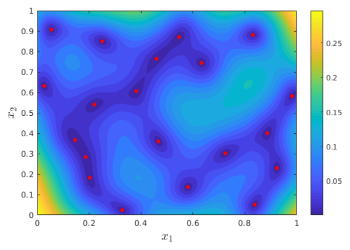
Visual comparisons are not without their uses when judging the performance of a surrogate model, however they are not always possible, and it is inadvisable to rely on them as the sole form of validation in any case. Instead, several of the diagnostics discussed in section 4.3.2 are employed to more accurately measure the performance of the emulator in approximating Franke's function. In particular, $m = 50$ validation points were obtained via LHS to act as the validation set that underpins each of the diagnostics. The root mean squared error, D^{RMSE}, of the emulator was found to be 0.0506. Figure 4.8a contains the individual prediction errors, D_i^I for each of the validation points, given by red dots. Each of the D_i^I lie within the bounds given by the dashed lines at -3 and 3 as expected with an accurate emulator. Finally, figure 4.8b showcases the correlation between the output of the true function and the output from the GPE. The diagonal dashed line represents the case where there is complete agreement between the two outputs. The red dots are the locations of the mean output, and the error bars represent the 2σ uncertainty bounds according to the posterior variance. On inspection, most of the points lie along the diagonal, and the respective uncertainty bounds contain the line for any points that fail to lie directly on it. The latter may indicate a lack of training data near to the validation point, resulting in a slight prediction error and larger uncertainty bounds within the nearby region of the input space.



(a)

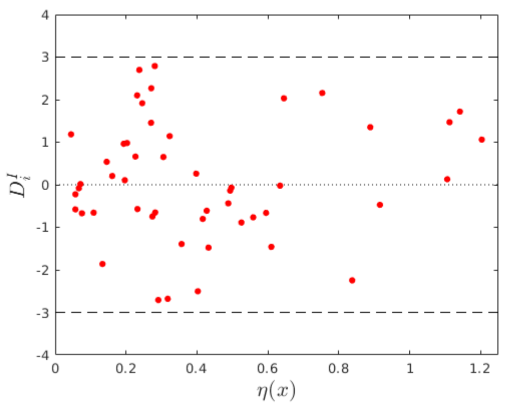


(b)

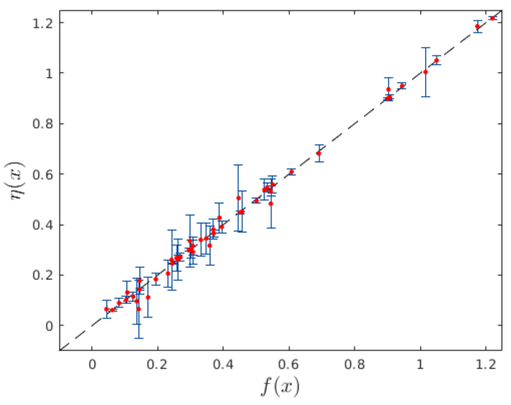


(c)

Figure 4.7: **Emulation of Franke's function** (a) True function; (b) GPE mean output and training data; (c) GPE variance and training data. In each, the respective values scale from purple (low) to yellow (high).



(a)



(b)

Figure 4.8: **GPE Diagnostics for Franke Function:** (a) individual prediction errors, (b) correlation between predictions and true values.

Overall, the GPE emulator managed to produce an accurate approximation to Franke's function, along with a realistic measure of uncertainty throughout the input domain. The main concepts of Gaussian process emulation were showcased to provide visual reinforcement of the earlier theory.

4.5 Enhancements of Gaussian Process Emulation

The preceding sections in this chapter provided the theoretical background and steps involved in utilising Gaussian process emulation for the purposes of approximating a computational expensive model. This approach has enjoyed widespread application and success throughout a multitude of fields of study. One of the by-products of this popularity is the emergence of a number of supplementary methods that can be considered enhancements of the GPE method. This section provides an overview of a number of these enhancements.

4.5.1 Adaptive Sampling

One of the main influences on the ability of a GPE to accurately portray the behaviour of the model it is attempting to emulate is the quality of the training set it is constructed on. Until this point, the construction of a GPE was only considered using training data obtained via some form of random sampling. However, the most popular enhancement of the GPE method is to employ adaptive sampling methods to produce a number of the training points. This is due to the fact that certain training points infer more information on the overall behaviour of a system than others. Such points are said to be located in areas of interest, for example a point corresponding to a local optimum. The goal of adaptive sampling is to attempt to obtain a higher proportion of the training points located within these areas of interest than would be the case when employing solely random sampling. To facilitate this, a GPE is typically constructed using an initial batch of randomly sampled data, and then points added iteratively through some adaptive sampling scheme until the computational budget is exhausted. A variety of distinct adaptive sampling methods have arisen over the years, each with their own characteristics, but a commonality is the task of maximising a utility function which measures some form of model improvement. The nature of this utility function is usually heavily dependent on the motivation behind the particular adaptive sampling scheme. Several of the most popular approaches are discussed below.

When the main motivation is to improve the approximation of the GPE throughout the entire input space, a common adaptive sampling scheme is to select points possessing high values of uncertainty [111]. More specifically,

$$\mathbf{x}_{AS} = \underset{\mathbf{x}}{\text{Maximise}}\; V^*(\mathbf{x}, \mathbf{x}), \tag{4.26}$$

where $\mathbf{x}_{AS}$ is the adaptively sampled point and $V^*(\mathbf{x}, \mathbf{x})$ is the posterior predictive covariance of the GPE from Eq. 4.25. Adding the point with the highest measure of uncertainty into the training set not only reduces the uncertainty to zero at the point itself, but can have a significant impact on both the uncertainty and mean approximation at other points within the vicinity, and thus aligning the emulator closer to the true model.

In certain problems, it is more desirable to increase the accuracy of the emulator at certain locations, rather than the accuracy of the emulator over the entire input space. For example, in optimisation problems, it makes more sense to sample from regions that may correspond to a global optimal point, than to sample from an area with suspected non-optimal performance, albeit with a higher degree of uncertainty. For the case where minimising the output of the model is the goal, a simple approach that utilises both the mean and covariance of the GPE is to minimise a lower confidence bound [112]

$$\mathbf{x}_{AS} = \underset{\mathbf{x}}{\text{Minimise}}\; LCB(\mathbf{x}) = M^*(\mathbf{x}) - aS(\mathbf{x}), \tag{4.27}$$

where $M^*(\mathbf{x})$ is the posterior predictive mean function from Eq. 4.16, $S(\mathbf{x}) = \sqrt{V^*(\mathbf{x}, \mathbf{x})}$ is the standard deviation of the GP output, and a is a scalar constant that controls the balance between exploration and exploitation. Increasing a increases the influence of the uncertainty on the selection process, encouraging exploration, while decreasing a encourages more exploitation.

A popular utility is to measure the improvement of a prospective adaptive sample. The improvement that a prospective sample offers is generally defined as $I(\mathbf{x}) = \max(y_{min} - M^*(\mathbf{x}), 0)$. This has led to two popular methods: probability of improvement (PI) and expected improvement (EI) in order to locate promising samples. The PI criterion attempts to find the location, where the probability of improving the objective function based on the current surrogate model is the highest [139]. The PI criterion is defined as

$$\mathbf{x}_{AS} = \underset{\mathbf{x}}{\text{Maximise}}\; PI(\mathbf{x}) = P(M^*(\mathbf{x}) < y_{min}) = \Phi\left(\frac{y_{min} - M^*(\mathbf{x})}{S(\mathbf{x})}\right), \tag{4.28}$$

where y_{min} represents the current best performing objective value amongst the training data, $M^*(\mathbf{x})$ and $S(\mathbf{x})$ are defined as in the LCB, and $\Phi(\cdot)$ represents the cumulative density function of a standard Gaussian random variable. The expected improvement (EI) [114] is the expected value of the improvement, and is defined as

$$\begin{aligned}\mathbf{x}_{AS} =& \underset{\mathbf{x}}{\text{Maximise}}\ E[I(\mathbf{x})] \\ =& |y_{min} - M^*(\mathbf{x})|\Phi\left(\frac{y_{min} - M^*(\mathbf{x})}{S(\mathbf{x})}\right) + S(\mathbf{x})\phi\left(\frac{y_{min} - M^*(\mathbf{x})}{S(\mathbf{x})}\right).\end{aligned} \tag{4.29}$$

where $\phi(\cdot)$ represents the probability density function of a standard Gaussian random variable, and the other terms are defined as in the other methods. EI in particular is extremely popular, and provides a good balance between exploration and exploitation of the input domain.

4.5.2 Gaussian Process Emulation with uncertain inputs

Section 4.2.2 described the steps involved in constructing a GPE which provides terms for the predictive posterior mean and predictive posterior covariance at any point $\mathbf{x}^*$. This point $\mathbf{x}^*$ can be known or unknown, depending on whether it is in the training set, but is considered certain, or noise-free. However, in order to perform robust optimisation, consideration needs to be made of the predictive output within a defined region surrounding $\mathbf{x}^*$ to reflect uncertain, or noisy, inputs. The most straightforward approach is to employ the RSS algorithm described in chapter 3 in conjunction with the GPE. However, in the case that the uncertainty within the input is normally distributed, that is $\mathbf{x}^* \sim \mathcal{N}(\mathbf{u}, \Sigma_x)$, it is possible to incorporate this input uncertainty directly into the GPE framework. This would allow for a standard optimisation algorithm to be used, and be theoretically much more efficient. To make a prediction at $\mathbf{x} \sim \mathcal{N}(u, \Sigma_x)$, the predictive distribution $p(y|\mathcal{D}, \mathbf{x})$ needs to be integrated over the input distribution

$$p(y|\mathcal{D}, \mathbf{u}, \Sigma_x) = \int p(y|\mathcal{D}, \mathbf{x})p(\mathbf{x}|\mathbf{u}, \Sigma_x)d\mathbf{x}. \tag{4.30}$$

However, as $p(y|\mathcal{D}, \mathbf{x})$ is a nonlinear function of $\mathbf{x}$, the updated predictive distribution $p(y|\mathcal{D}, \mathbf{u}, \Sigma_x)$ is not Gaussian, meaning that the integral can only be solved using approximation methods. The work in [115, 116, 117] describes the process involved to do this using both numerical and analytical methods. In particular, it was shown that the mean and variance of $p(y|\mathcal{D}, \mathbf{u}, \Sigma_x)$ could be extracted through either approximate moments or exact moments, depending on the nature of the kernel function utilised. The remainder of this subsection details the main steps involved in obtaining the exact moments based off employing a zero mean GPE with the squared exponential kernel. Further detail on this and the other methods can be found in [115].

Ultimately, the updated predictive distribution can be approximated as

$$p(y|\mathcal{D}, \mathbf{u}, \sigma_x) \approx \mathcal{N}(M(\mathbf{u}, \Sigma_x), V(\mathbf{u}, \Sigma_x)), \tag{4.31}$$

with updated mean and updated variance given by

$$M(\mathbf{u}, \Sigma_x) = \int M^*(\mathbf{x}) p(\mathbf{x}|\mathbf{u}, \Sigma_x) d\mathbf{x} \tag{4.32}$$

$$V(\mathbf{u}, \Sigma_x) = \int V^*(\mathbf{x}) p(\mathbf{x}|\mathbf{u}, \Sigma_x) d\mathbf{x} + \int M^*(\mathbf{x})^2 p(\mathbf{x}|\mathbf{u}, \Sigma_x) d\mathbf{x} - M(\mathbf{u}, \Sigma_x)^2, \tag{4.33}$$

where $M^*(\mathbf{x})$ and $V^*(\mathbf{x})$ are the mean and covariance from the noise-free case defined in Eq. 4.16 and Eq. 4.25 respectively. Alternatively, they can be defined as

$$M(\mathbf{u}, \Sigma_x) = \mathbb{E}[M^*(\mathbf{x})] \tag{4.34}$$

$$V(\mathbf{u}, \Sigma_x) = \mathbb{E}[V^*(\mathbf{x})] + \mathbb{E}[M(\mathbf{x})^2] - M(\mathbf{u}, \Sigma_x)^2 \tag{4.35}$$

Factoring in the fact that the GPE is zero-mean, and employing the notation of expectation gives

$$M(\mathbf{u}, \Sigma_x) = \gamma \mathbb{E}[t(\mathbf{x})] \tag{4.36}$$

$$V(\mathbf{u}, \Sigma_x) = \mathbb{E}[c(\mathbf{x}, \mathbf{x})] - (\mathbf{C}^{-1} - \gamma\gamma^T)\mathbb{E}[t(\mathbf{x})t(\mathbf{x})^T] - M(\mathbf{u}, \Sigma_x)^2, \tag{4.37}$$

where $\gamma = \mathbf{C}^{-1}\mathbf{y}$ is defined for simplicity, $\mathbf{t}(\mathbf{x}) = (c(\mathbf{x}, \mathbf{x}_1), ..., c(\mathbf{x}, \mathbf{x}_n))^T$ represents the correlation between $\mathbf{x}$ and the training data, and the expectation terms are defined as

$$\mathbb{E}[c(\mathbf{x}, \mathbf{x})] = \int c(\mathbf{x}, \mathbf{x}) p(\mathbf{x}|\mathbf{u}, \Sigma_x) d\mathbf{x} \tag{4.38}$$

$$\mathbb{E}[t(\mathbf{x})] = \int t(\mathbf{x}) p(\mathbf{x}|\mathbf{u}, \Sigma_x) d\mathbf{x} \tag{4.39}$$

$$\mathbb{E}[t(\mathbf{x})t(\mathbf{x})^T] = \int t(\mathbf{x})t(\mathbf{x})^T p(\mathbf{x}|\mathbf{u}, \Sigma_x)d\mathbf{x} \tag{4.40}$$

As discussed earlier, the correlation function utilised throughout this thesis is the squared exponential, defined as

$$c(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\bigg[-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T\psi^{-1}(\mathbf{x}-\mathbf{x}')\bigg]. \tag{4.41}$$

Alternatively, this can be represented as $c(\mathbf{x}, \mathbf{x}') = a\mathcal{N}_{\mathbf{x}}(\mathbf{x}', \psi)$, where $a = 2\pi^{\frac{d}{2}}\sigma^2|\psi|^{\frac{1}{2}}$. As a result,

$$\mathbb{E}[c(\mathbf{x}, \mathbf{x})] = \sigma^2 \tag{4.42}$$

$$\mathbb{E}[t(\mathbf{x})] = a\int \mathcal{N}_{\mathbf{x}}(\mathbf{x}_i, \psi)\mathcal{N}_{\mathbf{x}}(\mathbf{u}, \Sigma_x)d\mathbf{x} = a\mathcal{N}_{\mathbf{u}}(\mathbf{x}_i, \psi + \Sigma_x) \tag{4.43}$$

$$\begin{aligned}\mathbb{E}[t(\mathbf{x})t(\mathbf{x})^T] &= a^2\int \mathcal{N}_{\mathbf{x}}(\mathbf{x}_i, \psi)\mathcal{N}_{\mathbf{x}}(\mathbf{x}_j, \psi)\mathcal{N}_{\mathbf{x}}(\mathbf{u}, \Sigma_x)d\mathbf{x} \\ &= a^2\mathcal{N}_{\mathbf{x}}(\mathbf{x}_j, 2\psi)\mathcal{N}_{\mathbf{u}}\bigg(\frac{\mathbf{x}_i+\mathbf{x}_j}{2}, \Sigma_x + \frac{\psi}{2}\bigg)\end{aligned} \tag{4.44}$$

Substituting these values into Eq. 4.36 and Eq. 4.37 respectively and performing some rearranging gives

$$M(\mathbf{u}, \Sigma_x) = \gamma c(\mathbf{u}, \mathbf{x}_i)c_{corr}(\mathbf{u}, \mathbf{x}_i) \tag{4.45}$$

$$V(\mathbf{u}, \Sigma_x) = \sigma^2 - (\mathbf{C}^{-1} - \gamma\gamma^T)t(\mathbf{u})t(\mathbf{u})^T c_{corr2}(\mathbf{u}, \tilde{\mathbf{x}}) - M(\mathbf{u}, \Sigma_x)^2. \tag{4.46}$$

Here $\tilde{\mathbf{x}} = \frac{x_i+x_j}{2}$ and

$$c_{corr}(\mathbf{u}, \mathbf{x}) = |I + \psi^{-1}\Sigma_x|^{-\frac{1}{2}}\exp\bigg[-\frac{1}{2}(\mathbf{u}-\mathbf{x}_i)^T\Delta^{-1}(\mathbf{u}-\mathbf{x}_i)\bigg] \tag{4.47}$$

$$c_{corr2}(\mathbf{u}, \tilde{\mathbf{x}}) = |I + (\frac{\psi}{2})^{-1}\Sigma_{\mathbf{x}}|^{-\frac{1}{2}}\exp\bigg[-\frac{1}{2}(\mathbf{u}-\tilde{\mathbf{x}})^T\Omega^{-1}(\mathbf{u}-\tilde{\mathbf{x}})\bigg]. \tag{4.48}$$

where $\Delta^{-1} = \psi^{-1} - (\psi + \Sigma_x)^{-1}$ and $\Omega^{-1} = 2\psi^{-1} - (\frac{\psi}{2} + \Sigma_x)^{-1}$. As a result, prediction at an uncertain input is represented as the prediction at a certain point 'corrected' to factor in the input uncertainty. This 'correction' increases the correlation and generally decreases vertical amplitude.

A GPE extended to incorporate input uncertainty is referred to as a Robust GPE (RGPE) from this point onward. The function $f(x) = sin(x^2)$ was utilised as a test case to demonstrate the concept of the method. Both a noise-free GPE and a RGPE defined by $x \sim \mathcal{N}(u, 0.05)$ were trained with 15 training points obtained via LHS sampling. Figure 4.9 presents the results. The dashed black line denotes $f(x)$, the red dots the training data, and the blue line the mean output from the noise-free GPE. The green line represents the mean output from the RGPE. It is clear that the RGPE does not interpolate the training data as in the noise-free case. As a result, it cannot be used to approximate the true behaviour of the function, but instead to infer areas possessing a measure of robustness. Indeed, even though the vertical amplitude of the three local maxima of $f(x)$ are identical, as x increases, so does the impact of perturbation with the input values. Consequently, estimating the robust optimal of $f(x)$ is reduced to performing a standard optimisation procedure on the RGPE.

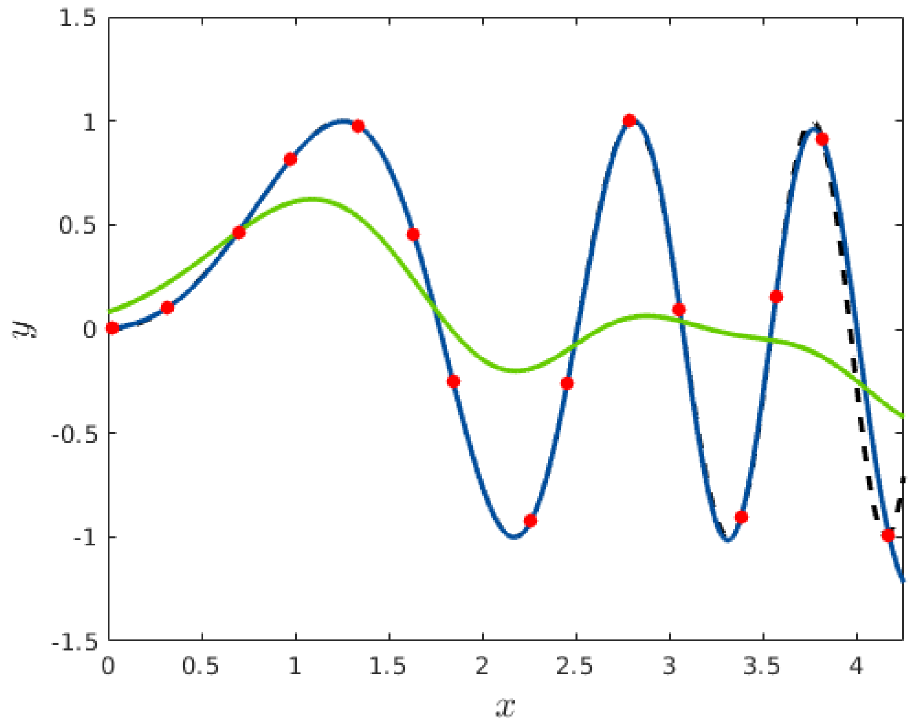


Figure 4.9: Mean output from a GPE (blue line) and a RGPE (green line) for the same function (black dashed line) and training data (red dots).

4.5.3 Multi-Fidelity Gaussian Process Emulation

The goal of a computational model is to provide an accurate approximation of the behaviour of a physical process within a spatial or temporal domain of interest. As highlighted in chapter 3, it is often the case that for a given physical process, a number of different computational models are available to describe its behaviour. These models can differ through a number of factors such as mathematical complexity, level of abstraction or model resolution. As a result, it is often possible to categorise the various models according to their respective computational cost and accuracy. This is commonly referred to as the level of fidelity of the computational model, with a lower fidelity (LF) model associated with a lower computational cost, but also less accuracy, while a higher fidelity (HF) model is associated with higher accuracy but at greater computational expense.

Within industrial applications, it is often critical that any computational model used represents the physical process with extremely high accuracy. This usually limits the use of any computational models to HF models only. However, such models are often limited to a low number of evaluations, due to a high computational cost. This problem provided the motivation for the earlier sections of this chapter, which describe the steps involved to construct a Gaussian process emulator in order to approximate some computational expensive model. However, as discussed in section 4.5.1, the performance of the GPE is highly influenced by the quality of the training set it is based on. Given that the LF model is describing the same physical process as the HF model, it can hypothetically provide information on the general behaviour of the process, but at a lower computational cost. For this reason, it is often desirable to utilise training data from both the HF model and the LF model, in order to maximise the information regarding the physical process for the given computational budget.

Gratiet and Garnier formulated a recursive approach to incorporate multi-fidelity (MF) data into the training of a GPE [119, 120] which this is adopted for use in this thesis. For a problem with s levels of ordered fidelity, the output for for the highest and most accurate level of fidelity is approximated by

$$\eta_s(\mathbf{x}) = \rho_{s-1}(\mathbf{x})\eta_{s-1}(\mathbf{x}) + \delta_s(\mathbf{x}). \tag{4.49}$$

Here $\eta_i(\mathbf{x})$ and $\delta_i(\mathbf{x})$ are GPEs, the subscripts denote the level of fidelity they are trained on and $\rho_i(\mathbf{x})$ is an adjustment function that attempts to 'correct' the lower-fidelity output to better represent the higher-fidelity output. The recursive approach is similar to the autoregressive approach developed by Kennedy and O'Hagan [118], and ultimately produces identical output. However, the autoregressive approach combines all information into a single formulation, which relies on inverting a $\sum_{i=1}^{s} n_i \times \sum_{i=1}^{s} n_i$ matrix, where n_i is the number of observations at the i^{th} level of fidelity. As the number of fidelities and training samples increase, inverting this matrix becomes extremely cumbersome and limits the effectiveness of the autoregressive approach. The recursive method on the other hand constructs s independent GPEs, with each trained using a different level of fidelity. By keeping the respective fidelities separate, the approach requires the inversion of s submatrices, which is less expensive and ill-conditioned than inverting the combined matrix used in the autoregressive approach. Moreover, constructing a distinct GPE for each fidelity simplifies the hyperparameter estimation task, and allows the user to analyse individual fidelities. For the case with two levels of fidelity, LF and HF, the recursive formulation is simply

$$\eta_{HF}(\mathbf{x}) = \rho_{LF}(\mathbf{x})\eta_{LF}(\mathbf{x}) + \delta_{HF}(\mathbf{x}), \tag{4.50}$$

with mean and covariance functions given by

$$\begin{aligned} M^*_{MF}(\mathbf{x}) = \rho(\mathbf{x})M^*_{LF}(\mathbf{x}) + h_{HF}(\mathbf{x})\beta_{HF} \\ + t^T_{HF}(\mathbf{x})C^{-1}_{HF}\big(\mathbf{y}_{HF} - \rho(\mathcal{D}_{HF}) \odot \mathbf{y}_{LF} - H_{HF}(\mathbf{x})\beta_{HF}\big) \qquad (4.51) \\ V^*_{MF}(\mathbf{x},\mathbf{x}') = \rho^2(\mathbf{x})V^*_{LF}(\mathbf{x},\mathbf{x}') + \sigma^2_{HF}\big(1 - c^T_{HF}(\mathbf{x},\mathbf{x}')\mathbf{C}^{-1}_{HF}c_{HF}(\mathbf{x},\mathbf{x}')\big) \qquad (4.52) \end{aligned}$$

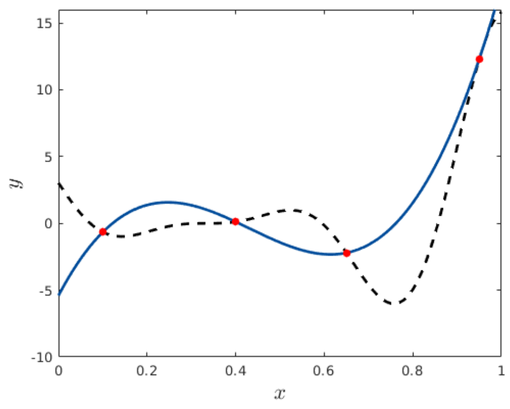
respectively. Here the subscript denotes the fidelity that a variable is associated with, for example $h_{HF}(\mathbf{x})$ represents the regression function associated with the GPE trained using HF data.

The Forrester functions [121] are commonly used to illustrate the impact of utilising multi-fidelity training data. The functions are defined as

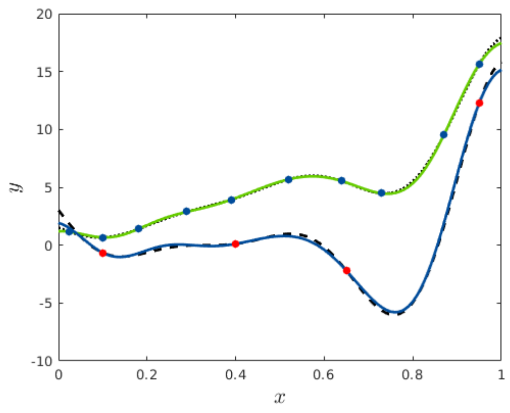
$$\begin{aligned} f_{HF}(x) &= (6x-2)^2\text{sin}(12x-4) \\ f_{LF}(x) &= \frac{1}{2}f_{HF}(x) + 10x \end{aligned} \tag{4.53}$$

where the HF and LF subscripts represent high-fidelity and low-fidelity respectively. To mimic a realistic engineering scenario, f_{HF} is limited to only four evaluations to provide for the HF training points. Figure 4.10 presents a comparison between approximating f_{HF} using a GPE trained solely on HF points, compared to also using MF training data. In each of the subfigures, the dashed black line represents the function the GPE is attempting to emulate, f_{HF}, while the HF training points are denoted by the red dots. Figure 4.10a illustrates the mean approximation (blue line) using a GPE trained solely on the four HF training points, denoted the HF GPE. Figure 4.10b shows the mean approximation of two separate GPEs. The green line represents the mean approximation from a GPE trained on LF training data, denoted by the blue dots. The goal of this GPE, referred to here out as the LF GPE, is to approximate f_{LF}, which is by dotted black line). The blue line represents the mean output from a GPE trained using MF data, i.e. the MF GPE. Comparing the two functions themselves, it is clear that they exhibit a similar shape albeit with some discrepancy between output values. On inspection, the LF GPE emulates f_{LF} almost perfectly, such that the only area that f_{LF} is visible at all is around $x = [0.95, 1]$. As a result, this provided information regarding the shape of the output space throughout the entire input domain, most notably illustrated by the fact that the MF GPE was able to recognise the dip located around $x = 0.75$, which the HF GPE missed. Indeed, the MF GPE provides a much better fit of f_{HF} than the HF GPE. Figure 4.10c provides the 2σ uncertainty bounds (light blue shaded regions) for the MF GPE, to highlight that the improvement of the mean approximation does not come at the cost of the ability to quantify the output uncertainty.

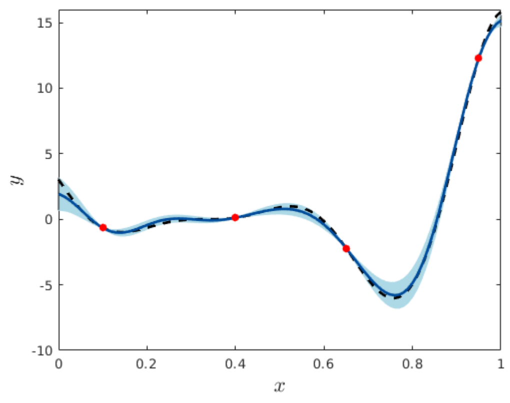
In general, most approaches to utilise MF training data work on the basis that each of the computational models available are modelling the same physical process, and that they can be categorised into nested levels of fidelity. However, it is possible to combine multiple computational models, even when the models are based on different physical processes, provided they possess some correlation between one another. This has been done through covariance-based methods that model between-output dependencies via a joint covariance function. Issues arise here with the notion of separability of the covariance function. Several papers adopt a separable covariance structure in which the correlation amongst inputs is the same for every output [122, 123, 124]. Although computationally efficient, there is a view that this is too restrictive and can inhibit emulator performance [125]. Several papers [125, 126, 127] construct a non-separable covariance structure using either convolution methods or the linear model of coregionalization to allow for mixing of different output



(a)



(b)



(c)

Figure 4.10: **Impact of MF training data:** The true HF function is given by the black dashed line and the HF trianing data by red dots in each of the subfigures. (a) shows the mean output of the HF GPE (blue line). (b) shows the LF function (black dotted line), LF training data (blue dots), LF GPE output (green line) and MF GPE output (blue line). (c) again depicts the MF GPE mean output, as well as its associated uncertainty bounds in shaded blue.

correlation lengths, addressing this concern. However, this adds a measure of complexity to the GPE method, and often possessing similar, or sometimes even worse, performance than two separate GPEs. As such, the work in this thesis limits any MF emulation to the situation where the computational models are approximating the same physical process, and there is a clear distinction between the levels of fidelity of the respective models.

4.6 Concluding Remarks of Chapter

The motivation of this chapter was to present an approach able to address the computational burden associated with working with computationally expensive models. Utilising several of the concepts discussed within this chapter to construct an accurate emulator, the RSS method described in chapter 3 can be applied directly without any computational issues, facilitating the robust optimisation of computationally expensive models. However, in the case that the uncertainty within the input variables follows a normal distribution, it is possible to combine several of the enhancements discussed within this chapter to formulate a more efficient robust optimisation procedure. This is discussed in greater detail in chapter 5.

Chapter 5

Multi-Fidelity Robust Gaussian Process Emulation[1]

This chapter presents a surrogate-based method, denoted MF-RGPE, to perform efficient robust optimisation of computationally expensive models. The method combines each of the enhancements discussed in chapter 4.5 to factor input uncertainty directly into the emulation process and allow for single-loop robust optimisation. The chapter begins by providing a mathematical description of the robust optimisation problem considering only input uncertainty, as well as factoring in the uncertainty associated with the emulator output. Next, a brief overview of some of the concepts and nomenclature for Gaussian process emulation and its enhancements are provided for the benefit of the reader. The proposed method is then discussed in detail. Finally a synthetic test problem is presented to showcase the method, before it is applied to two industrial case studies.

5.1 Introduction

The goal of engineering design is to create technological systems that satisfy specific performance objectives and constraints over a period of time. Usually, there exist many feasible designs that satisfy the required objectives. For this reason, it is desirable to choose an

[1]The results and ideas discussed in this chapter have been published in [128].

optimal design according to some criterion. Modern engineering systems are inherently complex. This complexity means that endogenous (geometry, material properties) and exogenous (loads) information is never complete, and often varies throughout the life cycle of the system (e.g. degradation altering geometry, etc). The objective of robust design is to determine a set of designs that exhibit high levels of performance with low variability, whilst taking uncertainties into account. The benefits of robust design include the assurance of high performance regardless of a variety of unknown factors and occurrences throughout the system's life cycle. Robust design is essentially a traditional optimisation task, but with an added constraint relating to the performance variability, or robustness, within some predefined neighbourhood of the input variables. There are various definitions of robustness. A detailed review of which is presented in [129], leading to various methodologies for tackling the robust optimisation problem. The authors of [130] employed a reliability-based optimisation algorithm which utilised Monte Carlo (MC) integration to obtain an averaged performance value within the neighbourhood. Similarly, [131] employed a probability distribution estimation method to obtain an approximate distribution of the performance within the neighbourhood. Another approach utilised the Taylor expansion of the expectation and variance of the performance and attempted to minimise both criteria simultaneously. Alternatively, several papers chose to optimise the worst-case scenario rather than any sort of averaged performance [132, 133].

Typically, the behaviour of modern engineering systems is modelled by computationally expensive simulators, which can be seen as mappings from the d-dimensional input space to the output space, denoted $f : \mathbf{x} \in \mathbb{R}^d \to y \in \mathbb{R}$. However, working directly with $f(\mathbf{x})$ is often infeasible due to computational expense. A popular approach to tackle this problem is to replace $f(\mathbf{x})$ with a surrogate model, which has been trained using data obtained from a low amount of simulator evaluations. One option is to train a Gaussian process emulator (GPE) [134, 135], which is defined by a mean function and a covariance function respectively. The mean function provides an inexpensive approximation to the simulator, $\eta(\mathbf{x}) \approx f(\mathbf{x})$, whilst the covariance function provides a measure of output uncertainty at each set of inputs, $V_x[\eta(\mathbf{x})]$ [39]. The result is that the robust design problem can be interpreted mathematically as

$$\begin{aligned}
&\underset{x}{\text{Minimise}} && \Big\{\eta(\mathbf{x}_\epsilon), V_x[\eta(\mathbf{x}_\epsilon)]\Big\}, \\
&\text{subject to} && g_j(\mathbf{x}_\epsilon) \geq 0, j = 1, 2, ..., m, \\
& && h_\nu(\mathbf{x}_\epsilon) = 0, \nu = 1, 2, ..., p.
\end{aligned} \tag{5.1}$$

Here $\mathbf{x}_\epsilon$ represents the set of input variables located within the hypercube, or neighbourhood, centered at $\mathbf{x}$ and bounded by $\mathbf{x} \pm \epsilon$. Consequently, $\eta(\mathbf{x}_\epsilon)$ represents the emulator output for this neighbourhood, and $V_x[\eta(\mathbf{x}_\epsilon)]$ a measure of uncertainty associated with the emulator output, whilst taking into account the added input uncertainty. Similarly, $g_j(\mathbf{x}_\epsilon)$ and $h_\nu(\mathbf{x}_\epsilon)$ are the respective inequality and equality constraints of this neighbourhood. In this context, robust design is interpreted as a double-loop optimisation task, with the outer-loop optimising the overall performance, subject to the constraint functions, and the inner-loop optimising for robustness in neighbourhood of the input variables (see algorithm 3).

Algorithm 3 Double-loop Optimisation

```
 1: Outer Loop
 2:     Minimise $\eta(\mathbf{x}_\epsilon)$, $V_x[\eta(\mathbf{x}_\epsilon)]$
 3:     Inner Loop
 4:     Generate $\mathbf{x}_i \in \mathbf{x}_\epsilon$, $i = 1, ..., N$
 5:         $\eta(\mathbf{x}_\epsilon) = \frac{1}{N}\sum_{i=1}^{N} \eta(\mathbf{x}_i)$
 6:         $V_x[\eta(\mathbf{x}_\epsilon)] = \frac{1}{N}\sum_{i=1}^{N} V_x[\eta(\mathbf{x}_i)]$
 7:         $g_j(\mathbf{x}_\epsilon)$ = Total Individual Violations of $g_j(\mathbf{x}_i)$
 8:         $h_\nu(\mathbf{x}_\epsilon)$ = Total Individual Violations of $h_\nu(\mathbf{x}_i)$
 9:     End
10: End
```

The ability of the GPE to accurately approximate $f(\mathbf{x})$ is directly related to the quality of the training set. This issue can be addressed through measures such as employing adaptive sampling schemes and supplementing the training set with data from multiple levels of fidelity. Adaptive sampling approaches tend to involve a utility function which measures some form of model improvement to select additional sample points. The most popular choice is expected improvement [114], which has been widely used in reliability [88], optimisation [136] and robust optimisation problems [137], amongst others. Further, the concept can be extended to multiple performance functions by considering the expected improvement of the current Pareto front via hypervolume expected improvement [138]. Other schemes include maximising the probability of improvement [139] or selecting samples with high uncertainty [140]. Multi-fidelity (MF) approaches are applicable when more than one potential simulator exists for the system under study. Lower-fidelity (LF) samples are defined by a lower-computational cost, but lower accuracy, than higher-fidelity (HF) samples. Multi-fidelity surrogate approaches exploit LF samples to gain information of the behaviour of the underlying system, and HF samples to maintain the desired accuracy. Most multi-fidelity approaches utilise LF data and adaptive sampling to attempt to sample the HF points in regions of interest and maximise the effectiveness of the surrogate [141, 142, 143]. Employing a surrogate model reduces the computational cost involved in robust design problems considerably. However, when there are a large number of performance functions and/or input variables, the double-loop approach becomes increasingly inefficient. A solution is to collapse the problem into a single-loop approach as done for a single-fidelity surrogate in [144]. In that paper, a GPE was enhanced to provide exact values of output uncertainty in the presence of uncertain inputs.

The work presented in this chapter provides a framework to perform efficient robust design on computationally expensive models. The framework adapts the single-loop approach discussed above to factor in multiple levels of fidelity, and supplements it with a hybrid adaptive sampling scheme. The chapter is organised as follows. Section 5.2 provides an overview of various forms of Gaussian process emulation. The proposed approach is introduced in Section 5.3 and discusses the main components. An illustrative example and two industrial CFD case studies are presented in Section 5.4. The final section provides relevant conclusions and highlights future work.

5.2 Methodology Overview

The main steps involved in each of the various forms of Gaussian process emulation are provided in detail in the last chapter, with some key remarks very briefly presented here for the benefit of the reader.

5.2.1 Single-Fidelity Gaussian Process Emulation

Given f is a computationally expensive mapping, under the Bayesian paradigm, $f(\mathbf{x})$ can be regarded as a random variable as the output is unknown until it is computed (and thus observed) by the modeller. Gaussian process emulation follows a Bayesian framework to provide a statistical approximation of $\eta(\mathbf{x}) \approx f(\mathbf{x})$. After initially assigning Gaussian Process prior and then updating using a series of training runs, ultimately the posterior distribution at some unobserved input x^*, conditional on the observed data, is given by [99]

$$\eta(\mathbf{x}^*)|\mathbf{D} \sim \mathcal{GP}(\mathbf{M}^*(\mathbf{x}^*), \mathbf{C}^*(\mathbf{x}^*, \mathbf{x}^{*\prime})), \tag{5.2}$$

with posterior predictive mean function

$$M^*(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*)^{\mathbf{T}}\beta + \mathbf{t}(\mathbf{x}^*)^{\mathbf{T}}\mathbf{\Sigma^{-1}}(\mathbf{y} - \mathbf{H}\beta), \tag{5.3}$$

and posterior predictive covariance function

$$\begin{aligned} C^*(\mathbf{x}^*, \mathbf{x}^{*\prime}) =& \hat{\sigma}^2\big[c(\mathbf{x}^*, \mathbf{x}^{*\prime}) + \mathbf{t}(\mathbf{x}^*)^{\mathbf{T}}\mathbf{\Sigma^{-1}}\mathbf{t}(\mathbf{x}^{*\prime}) \\ &+ (\mathbf{h}(\mathbf{x}^*)^{\mathbf{T}} - \mathbf{t}(\mathbf{x}^*)^{\mathbf{T}}\mathbf{\Sigma^{-1}}\mathbf{H})(\mathbf{H^T}\mathbf{\Sigma^{-1}}\mathbf{H})^{\mathbf{-1}} \\ &\times (\mathbf{h}(\mathbf{x}^*)^{\mathbf{T}} - \mathbf{t}(\mathbf{x}^*)^{\mathbf{T}}\mathbf{\Sigma^{-1}}\mathbf{H})^{\mathbf{T}}\big]. \end{aligned} \tag{5.4}$$

For the work in this chapter, the Gaussian process prior is assumed to have mean zero, i.e. $\mathbf{h}(\mathbf{x})^T = 0$, which will be adopted from here onward.

5.2.2 Multi-Fidelity Gaussian Process Emulation

Computationally expensive models are designed to capture the behaviour of an underlying physical system or product. As highlighted throughout this dissertation, it is often the case that more than one computational model is available, with models usually organised in levels of fidelity; a model with lower computational costs but less accuracy is considered to be of a lower fidelity than a more expensive and accurate model. As discussed in section 4.10, it is possible to approximate the output from the HF model as:

$$\eta_{HF}(\mathbf{x}) = \rho_{LF}(\mathbf{x})\eta_{LF}(\mathbf{x}) + \delta_{HF}(\mathbf{x}). \tag{5.5}$$

Here η_{LF} represents a GPE trained using data from the LF model, $\rho_{LF}(\mathbf{x})$ represents a regression function, and δ_{HF} represents a Gaussian Process Emulator which models the discrepancy between the HF estimation of $\rho_{LF}(\mathbf{x})\eta_{LF}(\mathbf{x})$ and the true HF simulator realisations. Both emulators are trained via the steps described in section (5.2.1). This can be generalised for t levels of fidelity in a recursive fashion:

$$\eta_t(\mathbf{x}) = \rho_{t-1}(\mathbf{x})\eta_{t-1}(\mathbf{x}) + \delta_t(\mathbf{x}). \tag{5.6}$$

5.2.3 Single-Fidelity Robust Gaussian Process Emulation

The two-looped approach to solving the robust optimisation problem (5.1) works by first attempting to minimise the objective functions $\eta(\mathbf{x})$ and $V_x[\eta(\mathbf{x})]$ in the outer-loop. Once a potential solution is found, the inner-loop measures the robustness over the input distribution. As a result, the predictive distribution of the emulator given input uncertainty is found by marginalising over the input distribution:

$$p\big(\eta(\mathbf{x}_*)|\mathbf{u}, \Sigma_x, D\big) = \int p\big(\eta(\mathbf{x}_*)|D\big)p\big(\mathbf{x}_*|\mathbf{u}, \Sigma_x\big)d\mathbf{x}_*. \tag{5.7}$$

This marginalisation is basically the aforementioned inner-loop and often achieved via MC sampling. In the case where the uncertainty within the inputs is normally distributed, i.e. for an unknown point $\mathbf{x}_* \sim \mathcal{N}(\mathbf{u}, \Sigma_x)$, it is possible to extract the first and second moments of (5.7) via methods described in section 5.2.3. These moments provide analytical expressions for the mean, $m(\mathbf{u}, \Sigma_x)$, and variance, $v(\mathbf{u}, \Sigma_x)$, of $p\big(\eta(\mathbf{x}_*)|\mathbf{u}, \Sigma_x, D\big)$. Ultimately, having direct access to the mean and variance of the emulator conditional on the input uncertainty collapses the robust optimisation problem down to a single-loop:

$$\begin{aligned} &\underset{u}{\text{Minimise}} && \Big\{m(\mathbf{u}, \Sigma_x), v(\mathbf{u}, \Sigma_x)\Big\}, \\ &\text{subject to} && g_j(\mathbf{u}, \Sigma_x) \geq 0, j = 1, 2, ..., m, \\ & && h_\nu(\mathbf{u}, \Sigma_x) = 0, \nu = 1, 2, ..., p. \end{aligned} \tag{5.8}$$

The resulting mean and variance functions are fundamentally equations (5.3) and (5.4) ‘corrected’ to factor in the input uncertainty. The added input uncertainty essentially flattens the output, with a decreased vertical amplitude and increased correlation.

5.3 Proposed Approach

The goal of the proposed approach is to perform efficient robust optimisation of computationally expensive models. The method is a combination of the various forms of Gaussian process emulation discussed in the previous section, and is termed Multi-Fidelity Robust Gaussian Process Emulator (MF-RGPE). When employing a GPE for the purposes of robust optimisation, the two main considerations are the ability of the GPE to accurately portray the behaviour of the underlying expensive model, and the efficiency of the robust optimisation process. To address the former, the proposed approach utilises training data from multiple levels of fidelity obtained via an extension of the Expected Improvement (EI) criterion [114] to maximise the quality of the training set. To increase the efficiency, the proposed approach extends the robust GPE detailed in section 5.2.3 to the multi-fidelity case. Further details of the steps are discussed in the following subsections.

5.3.1 Generating Training Samples

The framework begins with the design of experiment (DoE) of the LF model. Latin hypercube sampling (LHS) [85] is used as the space-filling algorithm to generate the initial samples. These samples are then evaluated on both the LF model and any relevant constraints functions, and are referred to as LF samples. To generate the initial HF samples, the LF samples are first sorted according to their objective and constraint values. A proportion of the top performing samples are selected to be part of the initial HF samples. The remaining initial HF samples are selected by filling the remaining space using a space-filling algorithm. This is done to encourage sampling of high-interest areas, whilst not neglecting the general performance of the GPE elsewhere. The proportion used in this work was 20% of initial samples from the top performing LF samples and 80% resulting from the space-filling algorithm. The HF samples were then evaluated on both the HF model and any relevant constraint functions.

5.3.2 Constructing the MF RGPE

The MF RGPE provides an approximation of the HF output whilst considering input uncertainty, and is constructed in a similar fashion to the standard MF GPE described in (5.5):

$$\eta_{RHF}(\mathbf{x}) = \rho_{RLF}(\mathbf{x})\eta_{RLF}(\mathbf{x}) + \delta_{HF}(\mathbf{x}). \tag{5.9}$$

Here, $\eta_{RLF}(\mathbf{x})$ represents a robust GPE trained using data from the LF model via the steps described in section 5.2.3 and $\rho_{RLF}(\mathbf{x})$ represents a regression function. The last term, $\delta_{HF}(\mathbf{x})$, represents a GPE which models the discrepancy between the estimation of the output of the HF training data, without accounting for input uncertainty, i.e. $\rho_{LF}(\mathbf{x})\eta_{LF}(\mathbf{x})$, and the actual HF simulator output. In an industrial context, there will usually be a predetermined computational budget and the stopping criterion will be met once this budget is exhausted. Other stopping criterion may include reaching a certain threshold of performance, such as obtaining a suitable design or reducing overall GPE uncertainty below some required value.

5.3.3 Adaptive Sampling

For a SF GPE, the expected improvement (EI) [114] at some point $\mathbf{x}$ is defined as

$$\mathbb{E}[I(\mathbf{x})] = |y_{min} - \eta(\mathbf{x})|\Phi\left(\frac{y_{min} - \eta(\mathbf{x})}{s}\right) + s\phi\left(\frac{y_{min} - \eta(\mathbf{x})}{s}\right). \tag{5.10}$$

Here y_{min} represents the current best performing objective value amongst the training data, s denotes the standard deviation of the GPE and $\Phi(\cdot)$ and $\phi(\cdot)$ represent the cumulative and probability density functions of a standard Gaussian random variable, respectively. EI attempts to locate samples that offer improved nominal performance against the current best sample. The method balances higher probability of a relatively small improvement (exploitation) versus a lower probability of high improvement (exploration). The concept of EI can also be applied to cases with more than one objective function by considering a hypervolume of improvement. Following the steps described in [145], given some reference point $\mathbf{r}$, the HVEI at some point $\mathbf{x}$ against a set of solutions $\mathbf{U}$ is defined as

$$HV\mathbb{E}[I(\mathbf{x})] = \sum_{i=1}^{N_U} \left[\prod_{j=1}^{N_{Obj}} \mathbb{E}\big[I(\mathbf{x}, U_{i,j})\big] \right], \tag{5.11}$$

where N_U is the number of points in the set U and N_{Obj} is the number of objective functions. Both EI and HVEI are designed for optimising nominal performance. The proposed approach extends them for the purposes of robust design by employing the mean and standard deviation GPE output from the SF RGPE and MF RGPE within the EI process. Consequently, the robust EI and robust HVEI can therefore be defined as

$$\mathbb{E}[I_R(\mathbf{x})] = |y_{Rmin} - \eta_{RMF}(\mathbf{x})| \Phi\left(\frac{y_{Rmin} - \eta_{RMF}(\mathbf{x})}{s_{RMF}}\right) + s\phi\left(\frac{y_{Rmin} - \eta_{RMF}(\mathbf{x})}{s_{RMF}}\right), \tag{5.12}$$

$$HV\mathbb{E}[I_R(\mathbf{x})] = \sum_{i=1}^{N_U} \left[\prod_{j=1}^{N_{Obj}} \mathbb{E}\big[I_R(\mathbf{x}, U_{i,j})\big] \right],$$

where y_{Rmin} is the current best performing objective value amongst the training data whilst also taking input uncertainty into account, and the RMF subscripts denote entities associated with the MF RGPE. Figure 5.1 illustrates the concept of robust HVEI in the case of two objective functions. The set of solutions, $\mathbf{P}$, represents the robust Pareto solutions taken from the current training data. Utilising these solutions and some reference point $\mathbf{r}$, a set of local upper bounds $\mathbf{U}$ can be constructed such that U_i lies at the intercept of P_i and P_{i+1}. The robust HVEI is thus the summation of the robust EI against each local upper bound, to give an overall value of improvement. To obtain promising adaptive samples, the single objective subset simulation algorithm described in section 2.3.2 is used to explore the input space and locate samples with high values of robust HVEI.

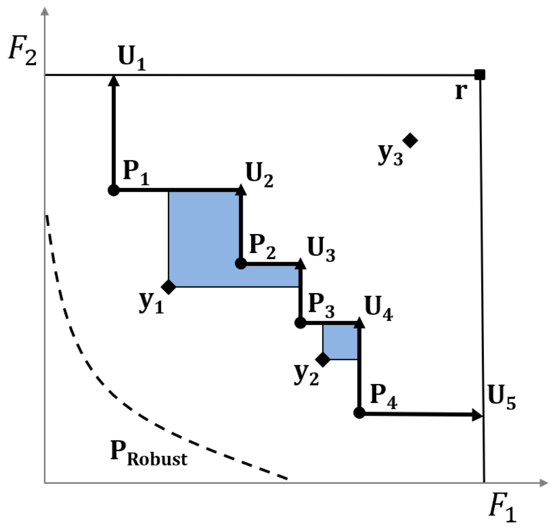


Figure 5.1: **Robust Hypervolume Expected Improvement:** The blue areas represent the hypervolume of improvement of a point y_i against the set $\mathbf{U}$. The point y_1, offers the most improvement, y_2 offers some improvement, while y_3 offers no improvement at all.

To increase efficiency, several samples are adaptively sampled in one optimisation iteration. An influence function [146], denoted $\tau(\mathbf{x})$, is employed to discourage the adaptive samples clustering in one area, by scaling the robust EI values after each new adaptive sample is taken, i.e.

$$\begin{aligned} \mathbb{E}[I_R(\mathbf{x})] &= \mathbb{E}[I_R(\mathbf{x})]\tau(\mathbf{x}), \\ \tau(\mathbf{x}) &= 1 - c(\mathbf{x}, \mathbf{x}_{AS}), \end{aligned} \tag{5.13}$$

where $\mathbf{x}_{AS}$ represents the latest adaptive sample and $c(\cdot,\cdot)$ is the correlation function from Eq. 4.41.

5.3.4 Robust Design

Once the computational budget is exhausted and the final batch of adaptive sampling completed, the final MF RGPEs can be utilised for robust design. Subset simulation is employed to locate the input regions corresponding to samples with high performance according to the MF RGPEs. These samples should be insensitive to perturbation in the values of the input variables, and given the computational resources, be validated on the HF model. The steps involved the the proposed method are outlined in a flowchart provided in figure 2. Steps 1-4 involve generating the LF and initial HF training data, and are described in section 5.3.1. This provides the foundation for the construction of the initial MF RGPE in step 5, which is detailed in section 5.3.2. Provided the stopping criterion (step 6) has not been met, this MF RGPE is then used as a tool to attempt to locate samples with improved performance in step 7, using the adaptive sampling process from section 5.3.3. This procedure is repeated on a loop, with an improved MF RGPE constructed at each generation until the stopping criterion is met. Optimisation of the MF RGPE(s) takes place in step 8.

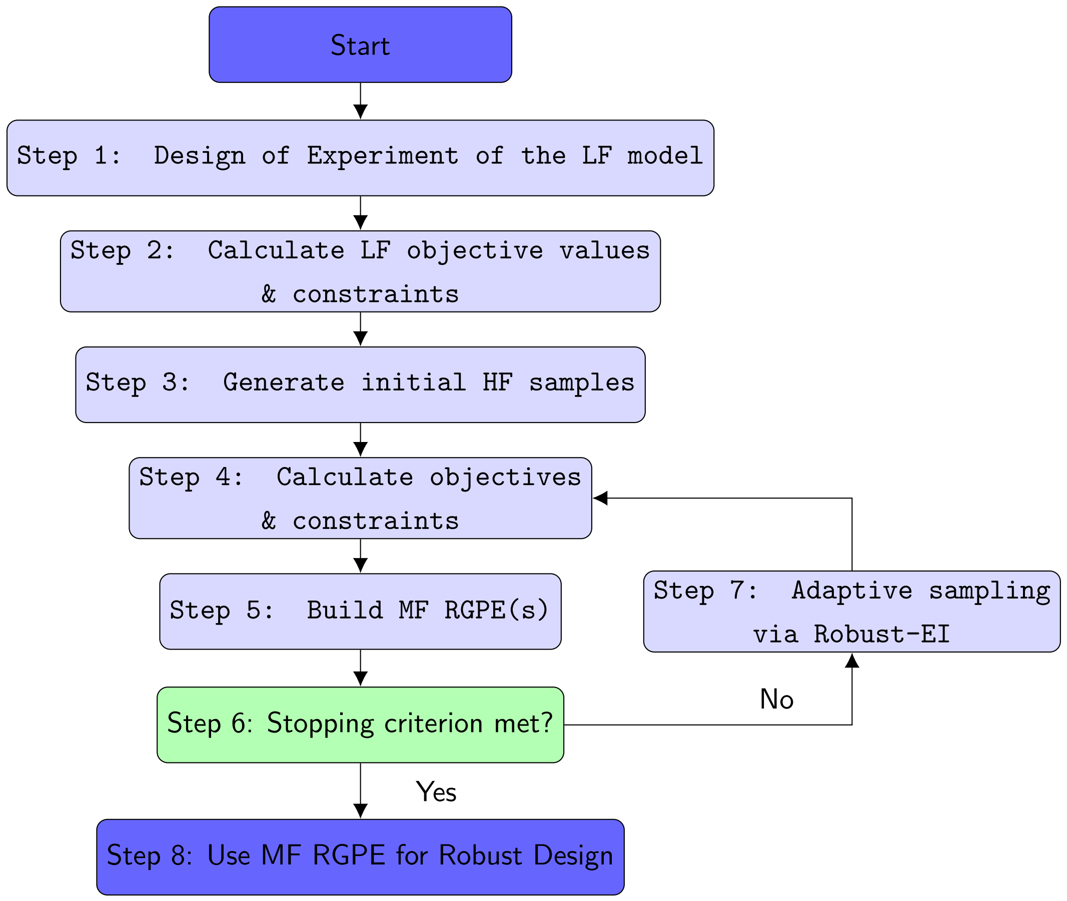

Start
Step 1: Design of Experiment of the LF model
Step 2: Calculate LF objective values & constraints
Step 3: Generate initial HF samples
Step 4: Calculate objectives & constraints
Step 5: Build MF RGPE(s)
Step 6: Stopping criterion met?
No
Step 7: Adaptive sampling via Robust-EI
Yes
Step 8: Use MF RGPE for Robust Design

Figure 5.2: Framework for Robust Design via multi-fidelity robust Gaussian process emulation.

5.4 Numerical Examples

This section provides three examples showcasing the MF RGPE approach discussed in the previous section. A synthetic example is first presented to showcase the concept of the approach before it is applied to two industrially-relevant test cases. In all examples, the regression function ρ_{RLF} from equation 5.9) is set to one, as in each example there is no assumed prior knowledge regarding the relationship between the LF simulator output and HF simulator output.

5.4.1 Synthetic Example

The motivation behind this synthetic example was to illustrate the main concepts of the proposed approach. The HF and LF functions are defined as

$$\begin{aligned} f_{HF}(x_1, x_2) &= \sin(x_1^2)\sin(x_2^2) + \frac{2(x_1+x_2)}{25}, \\ f_{LF}(x_1, x_2) &= \left(1+\frac{x_1}{5}\right)\sin\left(\frac{9x_1^2}{10}\right)\left(1+\frac{x_2}{5}\right)\sin\left(\frac{9x_2^2}{10}\right). \end{aligned} \tag{5.14}$$

The functions were constructed such that the LF function exhibited similar behaviour to the HF function, and as such could be used to infer regions of high interest. Additionally, both were designed to possess two maxima; a global maximum that was more sensitive to input uncertainty, and a local more robust maximum. The goal is to maximise f_{HF} in the face of some input uncertainty, with the intention to favour the more robust local maximum. An initial batch of 50 LF samples were selected via LHS. The 4 samples with the highest objective values were then selected, alongside 16 further samples from LHS, to populate the HF training set. A MF RGPE was then constructed with training data normalised between 0 and 1, and input uncertainty for an unknown point $\mathbf{x}_*$ is defined by the probability distribution $\mathbf{x}_* \sim \mathcal{N}(\mathbf{u}, diag[0.01, 0.01])$. Here $\mathbf{u}$ is the mean approximation of $\mathbf{x}_*$ while $diag[0.01, 0.01]$ is a diagonal matrix containing the variance with respect to each input variable. A further 3 samples were obtained via the robust EI adaptive sampling algorithm, and the retrained MF RGPE employed for robust optimisation. Finally, the inputs were transformed back to their original domains.

Figure 5.3 displays the contours of f_{LF} and f_{HF}, ax well as all samples for the MF RGPE.

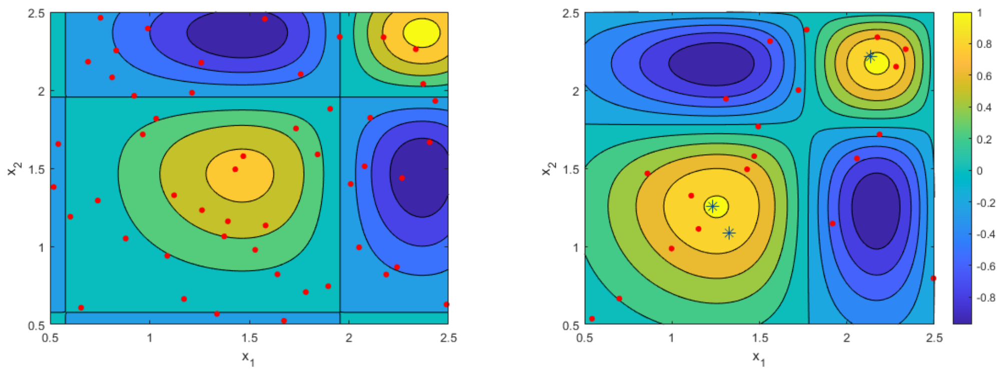


Figure 5.3: Left: Contour plot of LF function, LF samples denoted by the red dots. Right: Contour plot of HF function. The red dots are the initial 20 HF samples, whilst the blue stars represent those obtained via adaptive sampling.

On inspection, the function possesses a global optimum (maximum) located in the top right at approximately $\mathbf{x} \approx \left(\sqrt{(\frac{3\pi}{2})}, \sqrt{(\frac{3\pi}{2})}\right)$ and a local, more robust, optimum around $\mathbf{x} \approx \left(\sqrt{(\frac{\pi}{2})}, \sqrt{(\frac{\pi}{2})}\right)$ in the bottom left. The adaptive samples all lie within these regions of interest, with a preference to the local, more robust optimum. The local optimum in the bottom left is considered more robust as it has a wider base, meaning there is a lower drop in performance given any perturbation in the inputs. Note that several of the initial batch of HF samples were already in proximity to the two optima, highlighting the importance of utilising the best performing LF samples. Further, the LF data provided valuable information in the regions where HF samples were sparse (e.g. top left), saving an adaptive sample being wasted in an area of low interest. The illustrative example was repeated 10,000 times, and the normalised error from the true robust optimum presented in Figure 5.4. The error was normalised to illustrate the discrepancy between the true robust optimum and the actual values more clearly. The goal of the study was to showcase the individual steps described in Figure 2 and illustrate the merits of the approach. Overall, the majority of cases were within 1% of the true robust optimal input values.

5.4.2 Industrial Examples

Design engineers often utilise computationally expensive models in their design process. It is often desirable to factor input uncertainty into this process. The proposed approach has been designed to assist design engineers in this task, within a reasonable computational

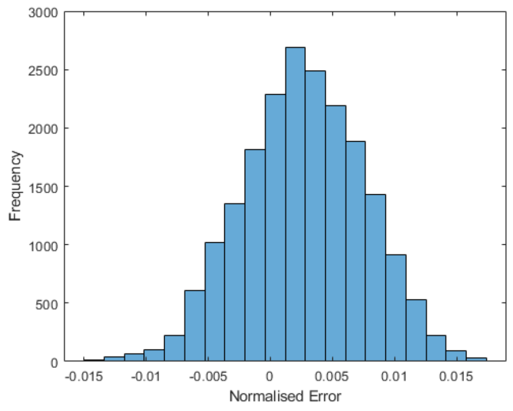


Figure 5.4: Histogram of normalised error: A comparison between the estimated robust optimal input values and the true robust optimal input values.

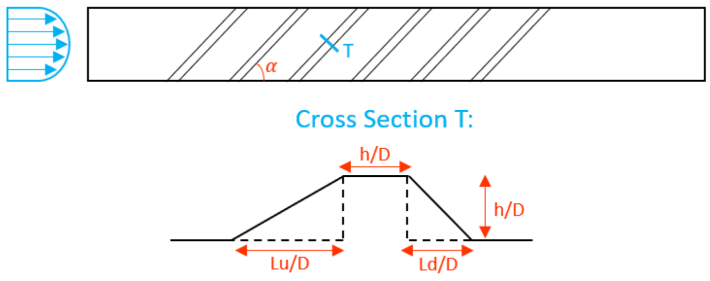
budget. Computational Fluid Dynamics (CFD) [15] models are a common tool in engineering design. They are usually computationally expensive, which limits their ability to be used directly in practical applications, but makes them a prime candidate for the MF RGPE approach.

Turbulated Duct Case Study

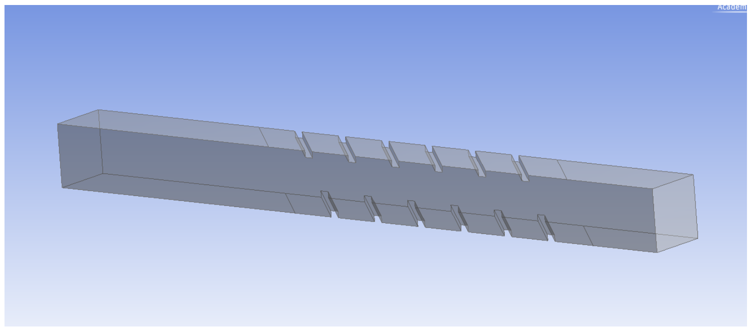
A frequent feature in turbine blades is the presence of turbulated internal cooling ducts. The presence of rib turbulators repeatedly perturb the boundary layer, which can result in significant heat transfer by promoting convective mixing with the core cooling flow. A downside is that this heat transfer comes at the cost of higher-pressure drop [147]. However, due to manufacturing constraints and degradation during the life cycle, the duct will likely diverge from the initial design at some point. The challenge is therefore to select a design that maximises heat flow, in this case Nusselt number, whilst minimising pressure drop in the face of input uncertainty. To address this challenge, a model of the turbulated duct was constructed using ANSYS software according to four geometric parameters that controlled the cross-sectional profile and angle of the turbulators, as shown in figure 5.5. The range of parameter values are shown in Table 5.1. Within the ANSYS software, each combination of these four parameters would result in a unique turbulated duct geometry. This geometry was then meshed using an unstructured tetrahedral grid and solved using Reynolds-averaged Navier–Stokes (RANS) [15] to output the Nusselt number and pressure coefficient for that particular design.

Turbulated Duct Input Variables		
Input Variable	Lower Bound	Upper Bound
α	35	55
Lu/D	0	0.5
Ld/D	0	0.5
h/D	0.05	0.15

Table 5.1: Input parameter ranges for turbulated duct case study.



(a)



(b)

Figure 5.5: **Turbulated Duct Geometry:** (a) Illustrates the influence the four geometric parameters have on the design. Additionally, the blue arrows to the left represent a mass flux profile mapped to the inlet to represent the operating conditions. (b) provides an example geometry taken from ANSYS.

For the MF RGPE approach, the overall computational budget assigned was equivalent to

44 HF samples. The LF model consisted of a mesh of approximately one million elements and solved using $k - \omega$ SST RANS on ANSYS, whilst the HF model consisted of a mesh of approximately five million elements and solved using $k - \omega$ SST RANS. The approximate computational cost was 20 LF samples $\approx$ 1 HF sample. Consequently, two separate MF RGPEs were trained for the Nusselt number and pressure coefficient respectively, with the initial MF RGPEs trained using 80 LF samples and 20 HF samples. A further 20 HF samples were adaptively selected in two batches of 10 samples to supplement the training set. The training data was normalised between 0 and 1, with input uncertainty represented via a diagonal matrix $\Sigma_x = diag[0.025, 0.025, 0.025, 0.025])$. The final MF RGPEs were then optimised using a multi-objective subset simulation algorithm. For comparison, the case study was repeated with the same computational budget, but using only HF samples to construct two SF RGPEs. The initial SF RGPEs were trained using 40 HF samples. A further 4 HF samples were adaptively selected in four batches of a single sample to supplement the training set.

Figure 5.6 demonstrates the adaptive sampling process and the final Pareto front for the MF RGPE approach. On inspection, several of the initial HF samples (green dots, top left plot) were located in close proximity to the eventual Pareto front, again highlighting the advantages of incorporating the LF training data to locate regions of interest. Indeed, the general performance of the adaptive points (blue stars) is significantly better than that of the randomly sampled points, showcasing the benefits of adaptive sampling. Furthermore, by comparing the two batches of adaptive sampling, it is clear how the adaptive sampling process attempts to converge towards the true Pareto front. It should be noted that the adaptive sampling procedure was assisted by the LF data to discard areas of low interest. The optimisation process placed constraints on the output variance of the respective GPEs to ensure a certain level of performance. This is highlighted in the close proximity between the Pareto front and the best performing training samples, placing further emphasis on the importance of quality training data. A single training point lies above the Pareto front, however this point was deemed to lack the necessary robustness according to the final MF RGPEs.

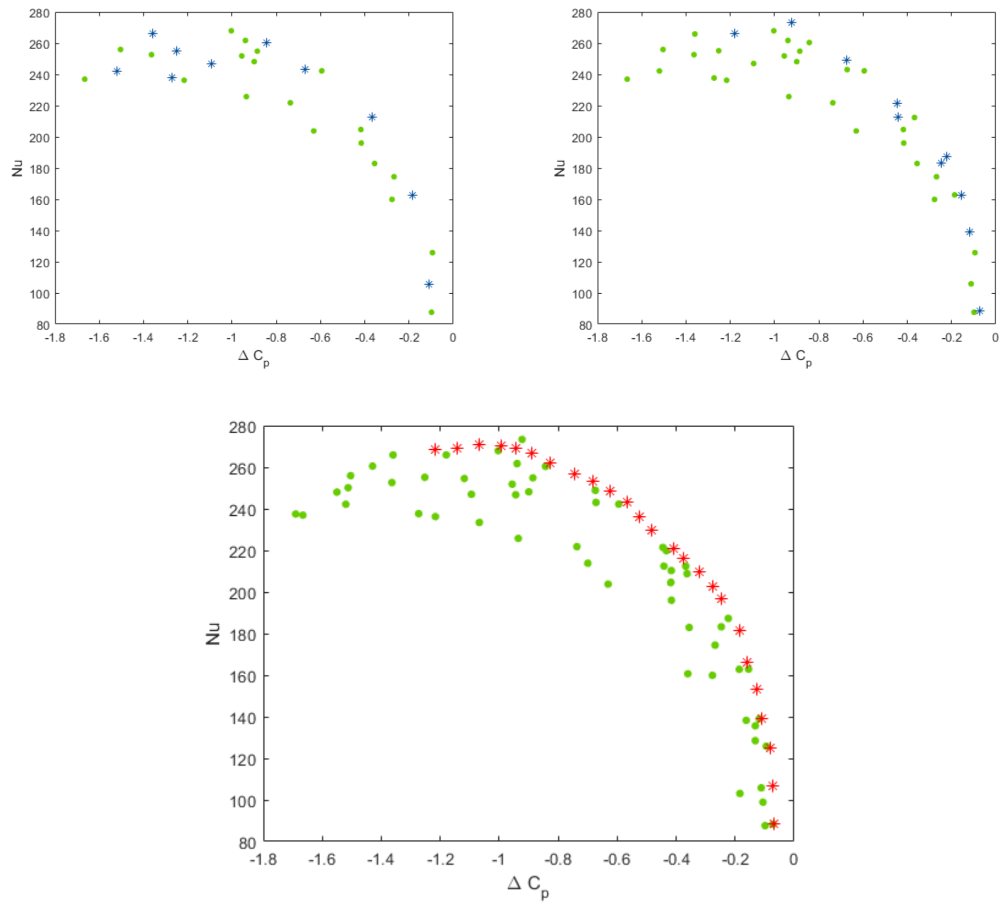


Figure 5.6: **HF Training Points and Pareto Front:** The top two figures showcase the adaptive sampling process, with the first batch top left and the second batch top right. In each, the green dots are the HF points that have previously been obtained, and the blue stars represent the newly evaluated adaptive samples. The plot on the bottom contains the full set of HF samples (green dots) and the Pareto front (red stars) obtained via optimising the MF RGPEs for each objective.

Figure 5.7 contains the Pareto fronts from the MF RGPE approach (red stars) and the SF RGPE approach (blue dots). In general, the two Pareto fronts possess similar behaviour, although the MF RGPE Pareto solutions exhibit superior performance than the SF RGPE Pareto solutions. A potential contributor to this discrepancy is the fact that the SF approach was made up of a higher proportion of randomly sampled training data. However, it is standard practice to use a budget of at least ten training samples per input dimension [86] in order to have sufficient confidence in the output of the underlying surrogate. The MF RGPE approach circumvents this issue by utilising LF data to make up for any loss of information. As such, it is reasonable that increasing the proportion of adaptively sampled data in the SF RGPE case would not necessarily improve the performance, due to surrogate inaccuracy and added uncertainty disrupting the sampling process. A second contributor is the fact that the MF RGPE approach was able to infer regions of high interest from the LF training data to aid in the adaptive sampling procedure. Overall, the MF RGPE offered superior performance than the SF alternative for the same computational budget.

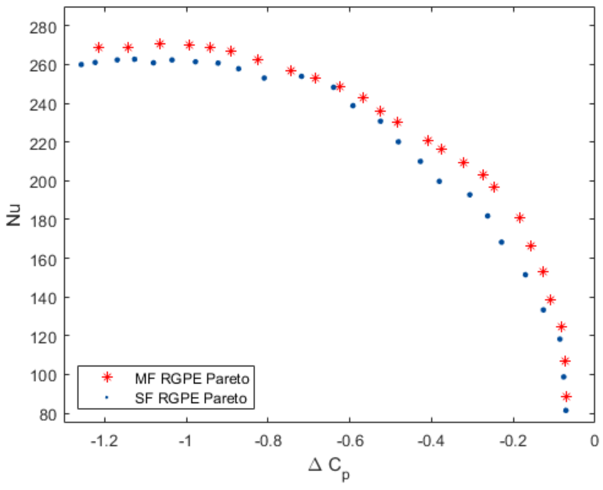


Figure 5.7: **Pareto Front Comparison:** The red stars represent the Pareto solutions obtained using the MF RGPE approach. The blue dots are the Pareto solutions obtained using the SF RGPE approach.

Figure 5.8 displays 20 validation samples for 4 designs taken from the MF RGPE Pareto front. A validation sample was generated from the distribution $\mathcal{N}(\mathbf{x}_P, \Sigma_x)$, where $\mathbf{x}_P$ denotes the original Pareto solution. The validation sample was then evaluated on the HF model to measure its performance. Given the computational cost involved, 4 Pareto solutions were chosen to ensure a reasonable amount of validation samples per Pareto solution could be evaluated, whilst making certain of validation across various areas of the Pareto front. It should be noted that the second and third Pareto solutions were located further from HF training samples than the other two solutions, hence their larger uncertainty bounds. Nevertheless, each validation sample was still within the 2σ uncertainty bounds of the original Pareto solution.

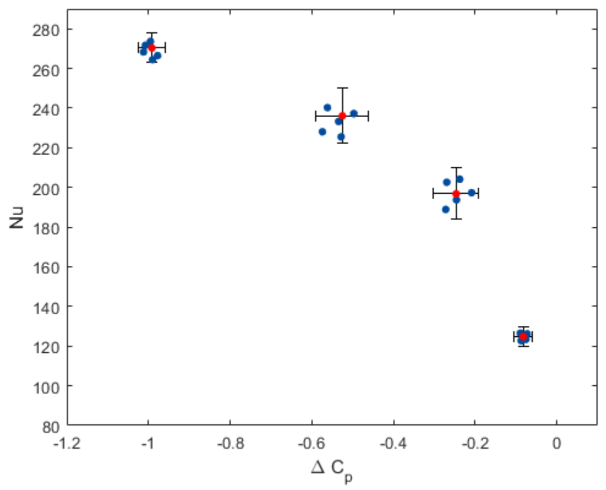


Figure 5.8: **Validation Points:** The red dots represent the original Pareto solutions and the black lines their respective uncertainty bounds. The blue dots are the simulator realisations at some perturbed design of the original Pareto solution.

Aerofoil Case Study

The aerofoil test case involved obtaining a set of aerofoil solutions that maximise lift-to-drag (L/D) ratio whilst minimising maximum blade thickness of a turbine blade in the face of potential perturbation of input values caused by uncertainty. A prospective aerofoil geometry was defined using the Class-Shape Transformation (CST) method [81]. In particular the Au and Al parameters are the weighting coefficients that help prescribe the thickness/shape at various locations along the upper and lower surfaces respectively. The parameters and their respective ranges are shown in Table 3.1. The LF model consisted of the aerofoil being solved over a range of angles of attack in XFOIL software, which performed a potential flow calculation without taking into account viscosity or a boundary layer. The HF model consisted of the aerofoil being solved via k-ω RANS in ANSYS. Unlike the turbulated duct case study, where the level of fidelity was solely due to mesh resolution, the fidelity in this case is dictated by two separate methods of varying accuracy and cost. It should be noted that the the definition of varying levels of fidelity is problem specific, with the only requirement that they exhibit similar behaviour in attempting to model the same underlying phenomena. The computational budget for the test case was approximately 240 HF samples. The comparative computational costs were approximately 20 LF samples per HF sample. Two separate MF RGPEs were trained for the lift-to-drag ratio and maximum thickness respectively, with the initial MF RGPEs trained using 600 LF samples and 120 HF samples. A further 80 HF samples were adaptively sampled in four batches of 20 to supplement the training set. The training data was normalised between 0 and 1, with input uncertainty represented via a 20×20 diagonal matrix Σ_x with each of the entries equal to 0.025. The final MF RGPEs were then optimised using a multi-objective subset simulation algorithm. As in the previous example, the case study was repeated using the same computational budget comprising of only HF samples. The initial SF RGPEs were trained using 200 HF samples, with a further four batches of a 10 samples added via adaptive sampling.

Figure 5.9 presents the adaptive sampling process and the final MF RGPE Pareto front. As in the turbulated duct study, several of the initial HF samples (green dots, top left plot) exhibited high performance and there was a clear convergence towards the suspected true Pareto front as the number of adaptive samples increased. The Pareto front closely followed the path of the best performing training samples. The training sample with the lowest maximum thickness was omitted from the Pareto front, as the performance of this point was particularly sensitive to input perturbations.

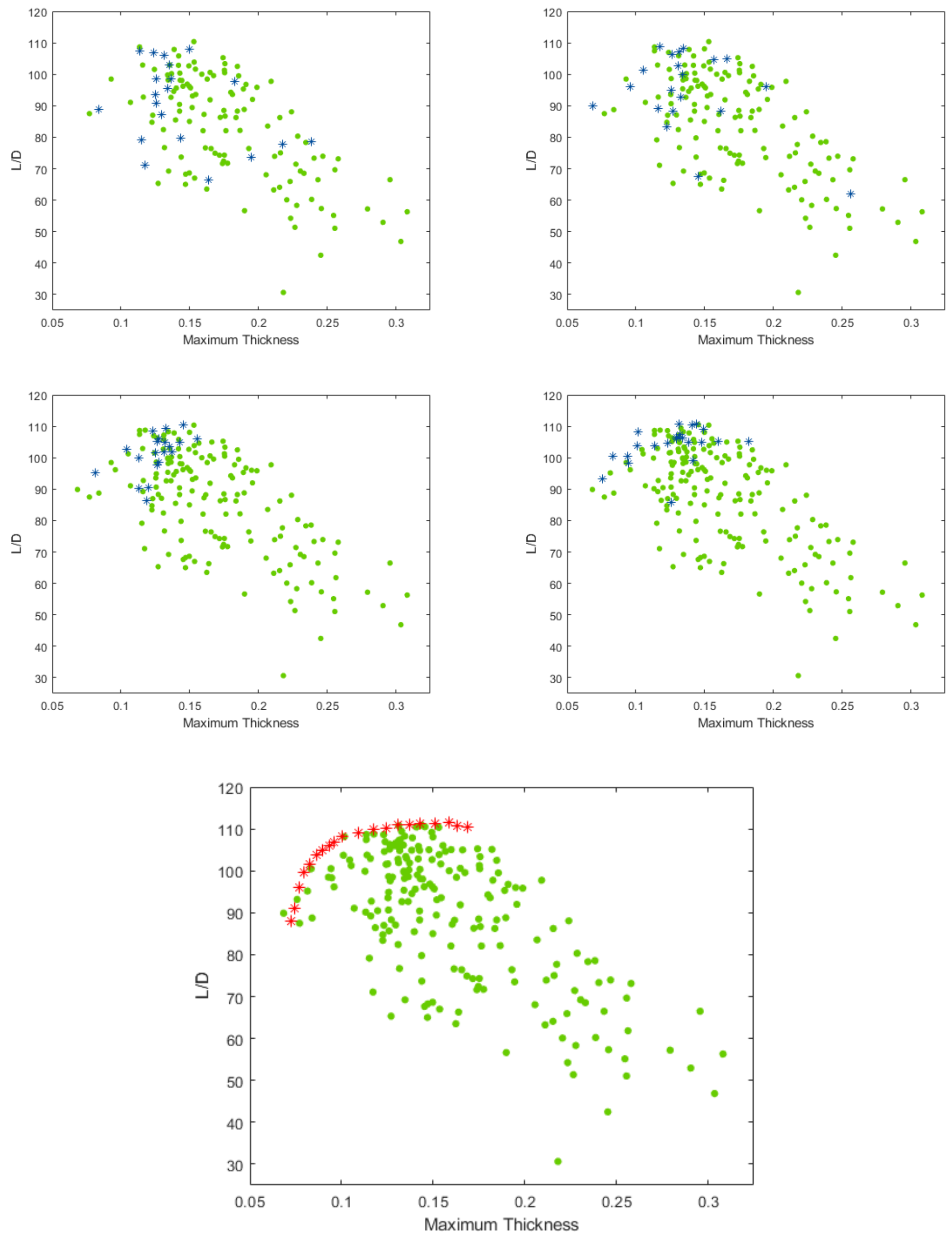


Figure 5.9: **HF Training Points and Pareto Front:** The top four figures contain the respective adaptive sampling batches. In order, top left, top right, bottom left, bottom right. The green dots are the HF points that have previously been obtained, and the blue stars represent the newly evaluated adaptive samples. The plot on the bottom contains the full set of HF samples (green dots) and the Pareto front (red stars) obtained via optimising the MF RGPEs for each objective.

Figure 5.10 contains the Pareto fronts from the MF RGPE approach (red stars) and the SF RGPE approach (blue dots). Both Pareto fronts initially rise relatively sharply before reaching a plateau with respect to the lift-to-drag coefficient. However, there is a significant discrepancy between the respective performance of the two Pareto fronts, with that of the MF RGPE completely dominating the SF RGPE counterpart. Whilst the SF RGPE approach wasted a number of samples searching in uncertain but ultimately low interest areas, the MF RGPE approach was able to discard these areas and target more promising locations due to the information provided by the LF data.

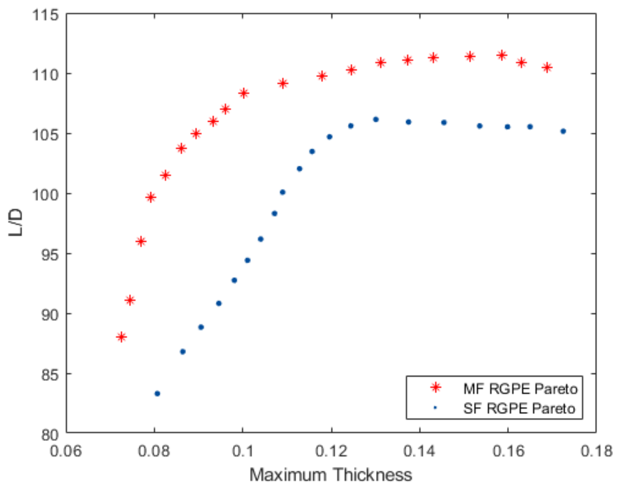
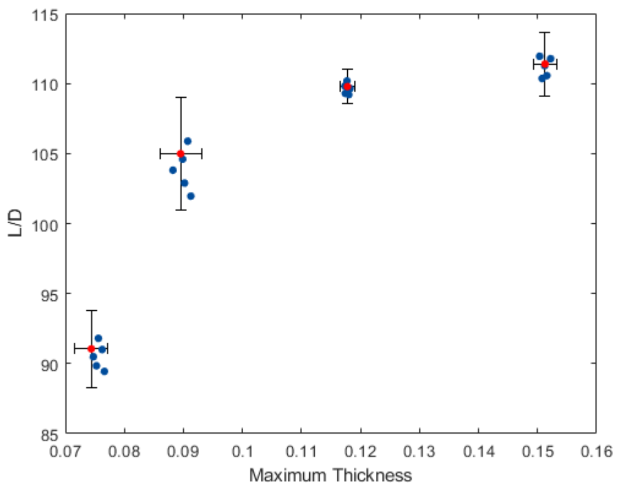


Figure 5.10: **Pareto Front Comparison:** The red stars represent the Pareto solutions obtained using the MF RGPE approach. The blue dots are the Pareto solutions obtained using the SF RGPE approach.

Figure 5.11 displays 20 validation samples for 4 designs taken from the MF RGPE Pareto front. As in the turbulated duct case study, the number of validation samples were limited due to the computational costs involved. The validation samples were selected to verify the performance of the MF RGPE approach across the Pareto front. It should be noted that there was zero discrepancy between the LF simulator and HF simulator output for the maximum thickness. As a result, there was significantly less GPE uncertainty for this objective, and the majority of the uncertainty bounds with respect to the maximum thickness is due to input uncertainty. Each validation sample was within the 2σ uncertainty bounds of the original Pareto solution. Moreover, the aerofoils corresponding to the validation points for the third (in ascending order of L/D ratio) Pareto solution were plotted against the original for a visual depiction of input uncertainty.



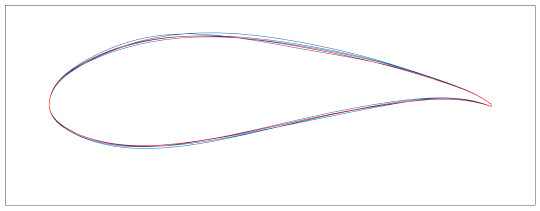


Figure 5.11: **Validation Points** Top: The red dots represent the original Pareto solutions and the black lines their respective uncertainty bounds. The blue dots are the simulator realisations at some perturbed design of the original Pareto solution. Bottom: Aerofoils for the validation samples (blue) and Pareto solution (red).

5.5 Concluding Remarks of Chapter

This chapter presented a emulation-based method, denoted MF RGPE, to perform efficient single-loop robust optimisation of expensive models. The motivation behind this method was to provide a flexible and reliable tool to facilitate robust optimisation of extremely expensive models, in the case where the computational constraint renders the MF-RSS method from chapter 3 infeasible. To maximise the effectiveness of the method, MF RGPE addresses the two main issues found when employing emulation-based approaches for robust optimisation, namely the quality of the emulator training set, and the efficiency of the robust optimisation process itself. Consequently, particular care was given to ensuring the highest quality of training data, through a combination of a novel adaptive sampling scheme and exploitation of multiple levels of fidelity. Further, factoring the input uncertainty directly into the emulation process negates the need for any neighbourhood sampling common in most robust optimisation algorithms. This collapses the optimisation portion down to a single-loop problem, significantly increasing the efficiency of the overall process.

Chapter 6

Conclusion

The research presented in this dissertation was focused on the development of tools and methods to tackle the challenge of performing robust optimisation of computationally expensive models. In particular, two approaches have been developed and tested against industrially relevant problems within the realm of computational fluid dynamics. The first approach is an extension of an MCMC algorithm known as subset simulation, that was developed to be applied directly to the computational model. The second approach is a combination of various enhancements of a surrogate modelling technique known as Gaussian process emulation. This approach factored input uncertainty into the construction of the surrogate model, which was then optimised in place of the actual computational model. Individual chapters offer a summary of individual results and contributions. The goal of this chapter is to act as a review of the work presented throughout this thesis, and to provide a direction for future research.

6.1 Summary of Completed Work

The ultimate goal of this thesis was to develop methods capable of performing optimisation of industrially relevant design problems, even in the case of uncertainty within the input parameters and computational constraints limiting the number of model evaluations available. Further, any methods needed to be conceptually simple to interpret, applicable to a wide array of problems and developed to require relatively minimal tuning or adaptation. Indeed to this end, the methods needed to be able to be deployed in a user-friendly MATLAB toolbox, to facilitate their use by the industrial partner's design engineers.

With this goal in mind, the first approach was an attempt to tackle the problem 'directly', and develop an optimisation algorithm that worked with the computational model directly. Chapter 2 presented a thorough overview of subset simulation, which was selected as an ideal candidate for the purposes of robust optimisation due to its straightforward concept and variety of useful properties. The chapter discussed the original motivation and concept behind the method, detailed the steps to adapt it for nominal optimisation and presented some test problems to highlight the algorithm in action. This work provided the foundation for the novel method presented in Chapter 3, which extended the applicability of the subset simulation algorithm to robust optimisation problems, by optimising over the averaged objective values for a neighbourhood surrounding a particular input value. Factoring input uncertainty into the optimisation problem addressed one of the challenges of the thesis, however more work was needed to address the second challenge of computational efficiency. To this end, and to ensure that the method was suitable for computationally expensive models, several measures were put in place. This included utilising an initial nominal optimisation stage in the algorithm to narrow the search to promising candidates, and avoid wasting resources on sampling from areas of low interest. This was followed by a robust optimisation stage, which reduced unnecessary model evaluations by storing all evaluated solutions in a *bank* of solutions, which could then be accessed to provide neighbourhood solutions without the need for new model evaluations. Further, for all uses of the subset simulation algorithm, an adaptive Markov Chain Monte Carlo algorithm was used to ensure optimum sample acceptance in order to boost efficiency. Finally, the method was also generalised to incorporate multiple levels of fidelity to further boost computational efficiency. Due to a lack of existing MF robust optimisation problems, this method, denoted MF-RSS was tested on a SF robust optimisation problems that was adapted to become a MF robust optimisation problem using a method found in a similar nominal optimisation problem within the literature. Lastly the method was successfully applied to an industrial case study provided by the academic partner.

As touched upon at several points during this dissertation, the complexity and computational cost of industrially relevant computational models can often render even the most efficient direct approaches infeasible. Consequently, this motivated the need for a second approach. Chapter 4 presented a thorough overview of Gaussian process emulation. The chapter began by introducing the notion of surrogate modelling, describing the benefits of utilising such methods when dealing with computationally expensive models. In particular, Gaussian process emulation was highlighted as a suitable surrogate method, due to its statistical nature providing a measure of uncertainty associated with its own predictions. The remainder of the chapter detailed the mathematical steps involved in the construction of a Gaussian process emulator, as well as several enhancements to improve its performance, namely utilising adaptive sampling schemes, factoring in uncertainty within the input variables and exploiting multi-fidelity training data. Employing this work in chapter 4 to produce the most accurate emulator possible, the RSS algorithm could then be applied to problems with computational constraints that would otherwise render it unusable. The use of an emulator would indeed add a new source of uncertainty to the problem, and as such it would be advisable to utilise the direct approach when computationally feasible. However, in the scenario that input uncertainty is normally distributed, chapter 5 discusses a Gaussian process emulation approach to perform efficient single-loop robust optimisation of expensive models, denoted MF RGPE. This method combines several enhancements of Gaussian process emulation to incorporate input uncertainty directly into the emulator output, allowing for the development of a novel robust adaptive sampling scheme, and increasing the efficiency of the overall robust optimisation process by facilitating the use of a nominal optimisation algorithm to perform robust optimisation. Due to the lack of existing suitable test problems, a test problem was developed to showcase the concept of the method before it was successfully applied to two industrial case studies.

6.2 Research Outlook

There are a number of potential areas of further research that can be considered based on the work discussed in this dissertation.

Chapter 3 described how uncertainty within a robust multi-objective problem could be generally categorised into one of three forms. The work in this thesis is solely concerned with robust problems of the first type, namely perturbations in the input variables. A natural development is to factor in the other forms of uncertainty that can arise in robust optimisation problems, namely the presence of noisy objective values and optimisation problems that change over time.

Another logical area of investigation is to apply the methods described in this dissertation to problems with more than two levels of fidelity. This will require some minor alterations to the methods in chapters 3 and 5, however the main concepts will remain unchanged. Additionally, with regards to the latter, another consideration is to extend the adaptive sampling portion of the approach to consider the level of fidelity as well as the location of prospective samples. Extending each of the methods to factor in a higher number of fidelities provides another potential area of further research. Within this thesis, the respective fidelities have been chosen using expert opinion. An area worth investigating is to develop a process that can guide the selection of the optimal number of fidelities, and their respective configurations.

Finally, for the Gaussian process based approaches, future work could include augmenting the present methods with measures to boost efficiency in higher-dimensional problems, such as employing principal component analysis or dimension reduction methods.

All of these developments are seen as potential basis for future collaboration with General Electric as they incorporate the current work into their in-house software.

6.3 Published Work

At the time of submission of this dissertation, the following work has been accepted for publication.

Ellison, M., Diaz De la O, F.A., Ince, N.Z., Willetts, M. (2021) ‘Multi-Fidelity Robust Optimisation using Subset Simulation’, *Applied Mathematical Modelling*, Vol. 100, pp. 92-106.

6.4 Work under Review

At the time of submission of this dissertation, the following work has been submitted and is currently under review.

Journal papers

M. Ellison, F.A. Diaz De la O, N.Z. Ince, M. Willetts, Multi-Fidelity Robust Optimisation using Subset Simulation, Under Review.

Appendix: Supplementary Subset Simulation Work

Algorithm 4 SuS Algorithm for Reliability

```
1: Set $p$, $n$.
2: $n_{chains}$=$np$
3: $n_{states}$=$\frac{1-p}{p}$
4: Generate $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ via MC
5: Evaluate objective values $\mathbf{Y} = f(\mathbf{X})$
6: Set $L = 0$, $n_F = 0$
7: **While** $\frac{n_F}{n} < p$
8:     Sort $\mathbf{X}, \mathbf{Y}$ by descending $Y$
9:     Set $n_{chains}$ top samples as seeds.
10:    Set $y_L^* = \frac{y_L^{np} + y_L^{np+1}}{2}$
11:    MCMC to produce candidate samples
12:    Accept candidate if $f(\mathbf{x}_{Cand}) > y_L^*$, reject otherwise
13:    Repopulate $\mathbf{X}$ using accepted MCMC samples
14:    Evaluate $\mathbf{Y}$
15:    Check $n_F$
16: **End**
17: Set $p_F = p^L \frac{n_F}{n}$
```

References

[1] Keane, A.J. and Nair, P.B. (2005) *Computational Approaches for Aerospace Design.* Chichester: Wiley.

[2] Turbocam International (no date) *Turbine Blades and Vanes.* Available at:¡ http://www.turbocam.com/blades¿ (Accessed: 17 June 2017).

[3] Taguchi, G. (1978) 'Performance analysis design', *International Journal of Production Research*, Vol. 16, No. 6, pp. 521. doi: 10.1080/00207547808930043.

[4] Taguchi, G. (1989) *Introduction to Quality Engineering.* American Supplier Institute

[5] Chen, W., Wiecek, M.M., and Zhang, J. (1999) 'Quality Utility - A Compromise Programming Approach to Robust Design', *Journal of Mechanical Design*, Vol. 121, No. 2, pp. 179–187.

[6] Ge.com (no date) *Powering Forward* Available at: https://www.ge.com/power [Accessed 26 September 2020].

[7] Ge.com (no date) *Gas Power Systems Gas Power Generation* Available at: https://www.ge.com/power/gas [Accessed 26 September 2020].

[8] Ge.com (no date) *GE Renewable Energy* Available at: https://www.ge.com/renewableenergy/wind-energy/offshore-wind/offshore-turbine-haliade-150-6mw [Accessed 26 September 2020].

[9] Keane, A. J. (2003) 'Wing Optimization Using Design of Experiment, Response Surface, and Data Fusion Methods', Journal of Aircraft, Vol. 40, No. 4, pp. 741-750.

[10] Chanzy, Q., and Keane, A. J. (2018) ‘Analysis and experimental validation of morphing UAV wings’, Aeronautical Journal, Vol. 122, No. 1249, pp. 390–408.

[11] Storti, B. A. et al. (2019) ‘Improving the efficiency of a Savonius wind turbine by designing a set of deflector plates with a metamodel-based optimization approach’, Energy, 186. doi: 10.1016/j.energy.2019.07.144.

[12] Keane, A.J. (2009) ‘Comparison of several optimization strategies for robust turbine blade design’ *Journal of Propulsion and Power* Vol. 25, No. 5, pp. 1092-1099.

[13] Hamza, K. and Saitou, K. (2012) ‘A co-evolutionary approach for design optimization via ensembles of surrogates with application to vehicle crashworthiness’, *Journal of Mechanical Design*, Vol. 134, No. 1, pp. 011001.

[14] Fan, X., Wang, P. and Hao, F. (2019) ‘Reliability-based design optimization of crane bridges using Kriging-based surrogate models’, *Structural and Multidisciplinary Optimization*, Vol. 59, No., pp. 993-1005. doi: 10.1007/s00158-018-2183-0.

[15] Versteeg, H.K. and Malalasekera, W. (2007) *An Introduction to Computational Fluid Dynamics: The Finite Volume Method.* Harlow: Pearson.

[16] Kenway, G.K.W. and Martins, J.R.R.A. (2014) ‘Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration’, *Journal of Aircraft*, Vol. 51, No. 1, pp. 144–160, doi: 10.2514/1.C032150.

[17] Campana, E.F. et al. (2006) ‘Shape optimization in ship hydrodynamics using computational fluid dynamics’, *Computer Methods in Applied Mechanics and Engineering*, Vol. 196, No. 1, pp. 634–651, doi: 10.1016/j.cma.2006.06.003.

[18] Booten, C.W. and Eaton, J.K. (2007) ‘Discrete Green’s Function Measurements in a Serpentine Cooling Passage’, *Journal of Heat Transfer*, Vol. 129, No. 12, pp. 1686–1696.

[19] Ekaterinaris, J.A, (2008), ‘High-order accurate space discretization methods for computational fluid dynamics’ in *Effective Computational Methods for Wave Propagation*, London: Chapman Hall/CRC, pp. 475-484

[20] Löhner, R. (2008) *Applied computational fluid dynamics techniques: an introduction based on finite element methods.* 2nd ed. England: John Wiley Sons.

[21] Mochimaru, Y. (1998) ‘Extension of spectral finite difference schemes for computational fluid dynamics’, Computers and Fluids, Vol. 27, no. 5, pp. 563–569. doi: 10.1016/S0045-7930(97)00062-5.

[22] Dick, E. (2009) *Introduction to finite volume methods in computational fluid dynamics.* Springer Berlin Heidelberg.

[23] Ferziger, J.H. and Peric, M. (2002) *Computational Methods for Fluid Dyanmics.* Berlin: Springer-Verlag.

[24] Goldstein, M. and Huntley, N., (2016) ‘Bayes linear emulation, history matching, and forecasting for complex computer simulators’ in *Handbook of Uncertainty Quantification*, Cham, Switzerland: Springer International Publishing, pp. 1–24.

[25] Kennedy, M., and O’Hagan, A., (2001) ‘Bayesian Calibration of Computer Models’, *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, Vol. 63, No. 3, pp. 425-464.

[26] Smith, R.C., (2013) *Uncertainty Quantification: Theory, Implementation, and Applications*, Philidelphia: SIAM.

[27] Pareto, V. (1896) *Cours D’Economie Politique, I and II*, Lausanne.

[28] Ansary, M.A.T., and Panda, G. (2020) ‘A Sequential Quadratic Programming Method for Constrained Multi-objective Optimization Problems’, *Journal of Applied Mathematics and Computing*, Vol. 64, No. 1–2, pp. 379-398. doi: 10.1007/s12190-020-01359-y.

[29] Povalej, Ž. (2014) ‘Quasi-Newton’s method for multiobjective optimization’, *Journal of Computational and Applied Mathematics.* Vol. 255, pp. 765-777.

[30] Kramer, O., (2017) *Genetic Algorithm Essentials.* Cham, Switzerland: Springer International Publishing.

[31] Kennedy, J. and Eberhart, R.C. (1995) ‘Particle swarm optimization’ *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia, pp. 1942-1948.

[32] Li, H. (2011), ‘Subset simulation for unconstrained global optimization’, *Applied Mathematical Modelling*, Vol. 35, No. 10, pp. 5108-5120.

[33] Beyer, H.G., and Sendhoff, B. (2007) ‘Robust optimization – A comprehensive survey’, *Computer Methods in Applied Mechanics and Engineering*, Vol. 196, No. 33, pp. 3190–3218. doi: 10.1016/j.cma.2007.03.003.

[34] Park, G.-J. (2007) *Analytic Methods for Design Practice*. London: Springer-Verlag.

[35] Hastie, T., Tibshirani, R. and Friedman, J. (2001) *The Elements of Statistical Learning*. New York: Springer.

[36] M. Abramowitz, I. Stegun, (1965) *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*, New York: Dover Publications.

[37] Powell, M.J.D., (1987) ‘Radial Basis Functions for Multi-variable Interpolation: A Review’ in: *Algorithms for Approximation*, Oxford: Clarendon Press, pp. 143–167.

[38] Vapnik, V.N. (1995) *The Nature of Statistical Learning Theory.* New York: Springer Verlag.

[39] O’Hagan, A. (2006) ‘Bayesian analysis of computer code outputs: A tutorial’. *Reliability Engineering and System Safety* Vol. 91, No. 10.

[40] Melchers, R. (1999) *Structural reliability analysis and prediction.* Chichester: John Wiley Sons.

[41] Madsen, H.O., Krenk, S., and Lind, N.C. (2006) *Methods of structural safety.* New York: Dover Publications.

[42] Hurtado, J.E., and Alvarez, D.A. (2003) ‘A classification approach for reliability analysis with stochastic finite element modeling’ *Journal of Structural Engineering*, Vol. 129 No. 8, pp. 1141–1149.

[43] Papadrakakis, M., Papadopoulos, V., and Lagaros, N.D. (1996) ‘Structural reliability analysis of elastic–plastic structures using neural networks and Monte Carlo simulation’. *Computer Methods in Applied Mechanics and Engineering*, Vol. 136, pp. 145–63.

[44] Schueller, G.I., Bucher, C.G., Bourgund, U., and Ouypornprasert, W. (1989). ‘On efficient computational schemes to calculate structural failure probabilities’. *Probabilistic Engineering Mechanics*, Vol. 4, No. 1, pp. 10-18.

[45] Koutsourelakis, P.S., Pradlwarter, H.J., and Schueller, G.I. (2004). ‘Reliability of structures in high dimensions, part I: algorithms and applications’. *Probabilistic Engineering Mechanics*, Vol. 19, No. 4, pp. 409–417.

[46] Engelund, S., and Rackwitz, R. (1993). ‘A benchmark study on importance sampling techniques in structural reliability’. *Structural Safety*, Vol. 12, No. 4, pp. 255–276.

[47] Au, S.K. and Beck, J.L. (2001) ‘Estimation of small failure probabilities in high dimensions by subset simulation’. *Probabilistic Engineering Mechanics*, Vol. 16, No. 4.

[48] Metropolis, N. (1987) ‘The beginning of the Monte Carlo method’, *Los Alamos Science*, Vol. 15, pp. 125

[49] Zuev, K. (2015) ‘Subset simulation method for rare event estimation: an introduction’. Available at http://arxiv.org/abs/1505.03506.

[50] Nataf, A. (1962). ‘Détermination des distributions de probabilité dont les marges sont donées’, *Comptes Rendues de l’Académie des Sciences*, Vol. 225, pp. 42-43.

[51] Rosenblatt, M. (1952) ‘Remarks on a multivariate transformation’, *The Annals of Mathematical Statistics*, Vol. 23, pp. 470-472.

[52] Zuev, K., Beck, J., Au, S. & Katafygiotis, L. (2012), ‘Bayesian post-processor and other enhancements of Subset Simulation for estimating failure probabilities in high dimensions’, *Computers And Structures* Vol. 92, pp. 283-296.

[53] Gilks, W.R., Richardson, S. and Spiegelhalter, D.J. (1998) *Markov chain Monte Carlo in practice*. Chapman Hall

[54] Santoso, A. M., Phoon, K.K. and Quek, S. T. (2011) ‘Modified Metropolis–Hastings algorithm with reduced chain correlation for efficient subset simulation’, Probabilistic Engineering Mechanics, 26(2), pp. 331–341. doi: 10.1016/j.probengmech.2010.08.007.

[55] Li, H. & Au, S. (2010), ‘Design optimization using Subset Simulation algorithm’, *Structural Safety*, Vol. 32, pp. 384-392.

[56] Dong, Y., Tang. J., Xu, B & Wang, D. (2005) ‘An application of swarm optimization to nonlinear programming’ *Comput Math Appl* Vol. 49, pp. 1655–68.

[57] Suo, X., Yu, X. and Li, H. (2017) ‘Subset simulation for multi-objective optimization’. *Applied Mathematical Modelling*, Vol. 44.

[58] Deb K., Pratap A., Agarwal S., Meyarivan T., (2002) ‘A fast and elitist multi-objective genetic algorithm: NSGA-II’, *IEEE Trans. Evol. Comput.* Vol. 6, No. 2, pp. 182–197.

[59] Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B., (2007) *Evolutionary Algorithms For Solving Multi-Objective Problems*, New York: Springer-Verlag.

[60] Ellison, M., Diaz De la O, F.A., Ince, N.Z., and Willetts M. (2021) ‘Multi-Fidelity Robust Optimisation using Subset Simulation’, *In preparation.*

[61] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) ‘Optimization by Simulated Annealing’ *Science* Vol. 220, No. 4598.

[62] Zhou, J. et al. (2017) ‘A multi-objective multi-population ant colony optimization for economic emission dispatch considering power system security’, *Applied Mathematical Modelling*, Vol. 45, pp. 684–704. doi: 10.1016/j.apm.2017.01.001.

[63] Hughes, TJR. (2000) *The Finite Element Method:Linear Static and Dynamic Finite Element Analysis.* New York, USA: Dover Publications.

[64] Zheng, L., Mittal, R., and Hedrick, T.L. (2013) ‘A multi-fidelity modelling approach for evaluation and optimization of wing stroke aerodynamics in flapping flight’, *Journal of Fluid Mechanics*, Vol. 721, pp. 118–154. doi: 10.1017/jfm.2013.46.

[65] Wiesmann, D., Hammel, U., and Back, T. (1998) ‘Robust design of multilayer optical coatings by means of evolutionary algorithms’, *IEEE Trans. Evol. Computat*, Vol. 2, No. 4, pp. 162–167. doi: 10.1109/4235.738986.

[66] Asafuddoula, M., Singh, H.K., and Ray, T. (2015) ‘Six-Sigma Robust Design Optimization Using a Many-Objective Decomposition-Based Evolutionary Algorithm’, *IEEE Trans. Evol. Computat*, Vol. 19, No. 4, pp. 490–507. doi: 10.1109/TEVC.2014.2343791.

[67] Rakshit, P., Konar, A., and Das, S. (2017) ‘Noisy evolutionary optimization algorithms – A comprehensive survey’, *Swarm and Evolutionary Computation*, Vol. 33, pp. 18–45. doi: 10.1016/j.swevo.2016.09.002.

[68] Jiang, S., and Yang, S. (2017) ‘A Steady-State and Generational Evolutionary Algorithm for Dynamic Multiobjective Optimization’, *IEEE Trans. Evol. Computat*, Vol. 21, No. 1, pp. 65–82. doi: 10.1109/TEVC.2016.2574621.

[69] Paenke, I., Branke, J., and Jin, Y. (2006) ‘Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation’, *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 4, pp. 405–420.

[70] Deb, K., and Gupta, H. (2006) ‘Introducing Robustness in Multi-Objective Optimization’, *Evolutionary Computation*, Vol. 14, No. 4, pp. 463–494.

[71] Meneghini, I.R., Guimaraes, F.G., and Gaspar-Cunha, A. (2016) ‘Competitive coevolutionary algorithm for robust multi-objective optimization: The worst case minimization’, 2016 IEEE Congress on Evolutionary Computation (CEC), p. 586.

[72] Hughes, M., Goerigk, M., and Dokka, T. (2020) ‘Particle swarm metaheuristics for robust optimisation with implementation uncertainty’, *Computers and Operations Research*, Vol. 122

[73] He, Z., Yen, G., and Lv, J. (2020) ‘Evolutionary Multiobjective Optimization With Robustness Enhancement’, *IEEE Transactions on Evolutionary Computation*, Vol. 24, No. 3, pp. 494–507.

[74] Papaioannou, I., Betz, W., Zwirglmaier, K. & Straub, D. (2015) ‘MCMC algorithms for Subset Simulation’, *Probabilistic Engineering Mechanics* Vol. 41, pp. 89-103.

[75] Au, S.K., Ching, J., and Beck, J.L. (2007) ‘Application of subset simulation methods to reliability benchmark problems’, *Structural Safety*, Vol. 29, No. 3, pp. 183–193. doi: 10.1016/j.strusafe.2006.07.008.

[76] Bourinet, J.-M., Deheeger, F., and Lemaire, M. (2011) ‘Assessing small failure probabilities by combined subset simulation and Support Vector Machines’, *Structural Safety*, Vol. 33, No. 6, pp. 343–353. doi: 10.1016/j.strusafe.2011.06.001.

[77] Hristov, P.O. et al. (2019) ‘Adaptive Gaussian process emulators for efficient reliability analysis’, *Applied Mathematical Modelling*, Vol. 71, pp. 138–151. doi: 10.1016/j.apm.2019.02.014.

[78] Ullmann, E, and Papaioannou, I. (2015) ‘Multilevel Estimation of Rare Events’, *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 3, No. 1, pp. 922–953

[79] Zitzler, E., Deb, K., and Thiele, L. (2000) ‘Comparison of Multiobjective Evolutionary Algorithms: Empirical Results’, *Evolutionary Computation*, Vol. 8, No. 2, pp. 173–195. doi: 10.1162/106365600568202.

[80] Zhou, Q. et al. (2019) ‘A two-stage adaptive multi-fidelity surrogate model-assisted multi-objective genetic algorithm for computationally expensive problems’, *Engineering with Computers*, p. 1-17, https://doi.org/10.1007/s00366-019-00844-8.

[81] Olsen, E., (2015). ‘Three-Dimensional Piecewise-Continuous Class-Shape Transformation of Wings’. In: *16th AIAA/ ISSMO Multidisciplinary Analysis and Optimization Conference*. Dallas, 22-26 June 2015. Institute of Aeronautics and Astronautics.

[82] Santner, T.J., Williams, B.J., and Notz, W.I., (2003) *The Design and Analysis of Computer Experiments*. New York: Springer-Verlag.

[83] Simpson, T. W. et al. (2001) ‘Metamodels for Computer-based Engineering Design: Survey and recommendations’, *Engineering with Computers: An International Journal of Simulation-Based Engineering*, Vol. 17, No. 2, pp. 129. doi: 10.1007/pl00007198.

[84] Pronzato, L., Müller, W.G., (2012) ‘Design of computer experiments: space filling and beyond’, *Statistics and Computing* Vol. 22, No. 3, pp. 681–701.

[85] McKay, M.D., Beckman, R.J., Conover, W.J., (1979), ‘Comparison of three methods for selecting values of input variables in the analysis of output from a computer code’, *Technometrics* Vol. 21, No. 2, pp. 239–245.

[86] Loeppky, J., Sacks, J. and Welch, W., (2009) ‘Choosing the Sample Size of a Computer Experiment: A Practical Guide’. *Technometrics*, Vol. 51, No. 4, pp.366-376.

[87] Owen, N. E. (2017) *A comparison of polynomial chaos and Gaussian process emulation for uncertainty quantification in computer experiments*, PhD thesis, University of Exeter.

[88] Hristov, P. O. (2018) *Numerical modelling and uncertainty quantification of biodiesel filters*, PhD thesis, University of Liverpool.

[89] Rougier, J. (2007) ‘Probabilistic inference for future climate using an ensemble of climate model evaluations‘ *Climate Change* Vol. 81, No.3.

[90] Kennedy, M.C., Anderson, C.W, Conti, S. and O’Hagan, A. (2006) ‘Case studies in Gaussian process modelling of computer codes’ *Reliability Engineering and Systems Safety* Vol. 91, No. 10.

[91] Kolachalama, V., Bressloff, N. and Nair, P. (2007) ‘Mining data from hemodynamic simulations via Bayesian emulation’ *Biomedical Engineering Online* Vol. 6, No. 47.

[92] Bates, R., Kennet, R., Steinberg, D. and Wynn, H. (2006) ‘Achieving robust design from computer simulations’ *Quality Technology and Quantitative Management* Vol. 3, No. 2.

[93] Khuri, A.I. and Mukhopadhyay, S. (2010), ‘Response surface methodology’ *Wiley Interdisciplinary Reviews: Computational Statistics* Vol. 2, No. 2.

[94] O’Hagan, A. (2011) ‘Polynomial Chaos: A Tutorial and Critique from a Statistician’s Perspective’, *SIAM/ASA J. Uncertainty Quantification*, Vol. 20, pp. 1–20.

[95] Matheron, G., (1963) ‘Principles of geostatistics’, *Economic Geology*, Vol. 58, pp. 1246–1266.

[96] Krige, D., (1951) ‘A statistical approach to some basic mine valuation problems on the Witwatersrand’, *Journal of Chemical, Metallurgical, and Mining Society of South Africa*, Vol. 52, pp. 119–139.

[97] Sacks, J., Welch, W.J., Mitchell, T.J., and Wynn, H.P. (1989) ‘Design and analysis of computer experiments’, *Statistical Science*, Vol. 4, No. 4, pp. 409–435.

[98] Cressie, N., (1990) *The origins of kriging, Mathematical Geology*, Vol. 22, No. 3, pp. 239–252.

[99] Oakley, J., (1999) *Bayesian uncertainty analysis for complex computer codes*, PhD thesis, University of Sheffield.

[100] C. M. Bishop (2006) *Pattern recognition and machine learning*, New York : Springer.

[101] Rasmussen, C.E., Williams, C.K.I., (2006) *Gaussian processes for machine learning*, Cambridge, Massachusetts, United States: MIT Press.

[102] MacKay, J.C., (1998) 'Introduction to Gaussian processes' in *Neural Networks and Machine Learning*, Amsterdam: Kluwer Academic Press, pp. 133–166.

[103] Haylock, R.G.E., (1997) 'Bayesian inference about outputs of computationally expensive algorithms with uncertainty on the inputs', PhD. thesis, University of Nottingham.

[104] Nelder, J. A., Mead, R., (1965) 'A simplex method for function minimization', *The Computer Journal*, Vol. 7, No. 4, pp. 308–313.

[105] Metropolis, N. et al (1953) 'Equation of State Calculations by Fast Computing Machines', *The Journal of Chemical Physics* Vol. 21, No. 6, pp. 1087–1092.

[106] Song, H., Choi, K.K., Lamb, D. (2013) 'A Study on Improving the Accuracy of Kriging Models by Using Correlation Model/Mean Structure Selection and Penalized Log-Likelihood Function', *10th World Conference on Structural and Multidisciplinary Optimization*, Orlando, pp. 1–10.

[107] Andrianakis, I, Challenor, P. (2012) 'The effect of the nugget on Gaussian process emulators of computer models', *Computational Statistics and Data Analysis*, Vol. 56, pp. 4215–4228.

[108] Ranjan, P., Haynes, R., Karsten, R. (2011) 'A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data', *Technometrics* Vol. 53, No. 4, pp. 366–378.

[109] Bostanabad, R. et al. (2018) 'Leveraging the nugget parameter for efficient Gaussian process modeling', *International Journal for Numerical Methods in Engineering*, Vol. 114, No. 5, pp. 501-516. doi: 10.1002/nme.5751.

[110] Bastos, L.S. and O'Hagan, A. (2009) 'Diagnostics for Gaussian process emulators', *Technometrics*, Vol. 51, No. 4.

[111] Ghosh, S. et al. (2019) 'A strategy for adaptive sampling of multi-fidelity Gaussian processes to reduce predictive uncertainty', *Proceedings of the ASME Design Engineering Technical Conference*, Anaheim, California, August 18-21, doi: 10.1115/DETC2019-98418.

[112] Forrester, A.I.J. and Keane, A.J. (2009) 'Recent advances in surrogate-based optimization', *Progress in Aerospace Sciences*, Vol. 45, No. 1, pp. 50–79. doi: 10.1016/j.paerosci.2008.11.001.

[113] Ruan, X., Jiang, P., Zhou, Q., Hu, J., Shu, L. (2020) 'Variable-fidelity probability of improvement method for efficient global optimization of expensive black-box problems'. *Struct Multidiscip Optim* pp. 1–32.

[114] Jones, D.R., Schonlau, M. and Welch, W.J. (1998) 'Efficient Global Optimization of Expensive Black-Box Functions'. *Journal of Global Opt.* Vol. 13 No.4, pp. 455-492.

[115] Girard, A. (2004) *Approximate Methods for Propagation of Uncertainty with Gaussian Process Models*, PhD thesis, University of Glasgow.

[116] Quinonero-Candela, J., Girard, A., and Rasmussen, C. E., (2003) 'Prediction at an uncertain input for Gaussian processes and relevance vector machines application to multiple-step ahead time-series forecasting'. Tech. rep., Technical University of Denmark, October.

[117] Quinonero-Candela, J., Girard, A., Larsen, J., and Rasmussen, C. E., (2003) 'Propagation of uncertainty in bayesian kernels models - application to multiple-step ahead forecasting'. *International Conference on Acoustics, Speech and Signal Processing*, April.

[118] Kennedy, M., and O'Hagan, A. (2000) 'Predicting the output from a complex computer code when fast approximations are available', *Biometrika*, Vol. 87, No. 1, pp. 1–13.

[119] Le Gratiet, L., and Garnier, J. (2014) 'Recursive co-kriging model for design of computer experiments with multiple levels of fidelity', *International Journal for Uncertainty Quantification*, Vol. 4, No. 5, pp. 365–386.

[120] Le Gratiet, L., (2004) *Multi-fidelity Gaussian process regression for computer experiments*, PhD thesis, Université Paris-Diderot.

[121] Forrester, A.I.J., Sóbester, A., and Keane, A.J., 'Multi-Fidelity Optimization via Surrogate Modeling', *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 463, No. 2088, 2007, pp. 3251–3269. doi:10.1098/rspa.2007.1900

[122] Bilionis, I. et al. (2013) 'Multi-output separable Gaussian process: Towards an efficient, fully Bayesian paradigm for uncertainty quantification', *Journal of Computational Physics*, Vol. 241, pp. 212–239.

[123] Conti, S. and O'Hagan, A. (2010) 'Bayesian emulation of complex multi-output and dynamic computer models', *Journal of Statistical Planning and Inference*, Vol. 140, pp. 640–651.

[124] Rougier, J. (2008) 'Efficient Emulators for Multivariate Deterministic Functions', *Journal of Computational and Graphical Statistics*, Vol. 17, No. 4, pp. 827-843.

[125] Fricker, T., Oakley, J. and Urban, N. (2013) 'Multivariate Gaussian Process Emulators With Nonseparable Covariance Structures', *Technometrics*, Vol. 55, No. 1, pp. 47-56.

[126] Zhang, B. et al. (2015) 'Full scale multi-output Gaussian process emulator with nonseparable auto-covariance functions', *Journal of Computational Physics*, Vol. 300, pp. 623–642.

[127] Wang, B. and Chen, T. (2015) 'Gaussian process regression with multiple response variables', *Chemometrics and Intelligent Laboratory Systems*, Vol. 142, pp. 159–165.

[128] Ellison, M., Diaz De la O, F.A., Ince, N.Z., Willetts, M. (2021) 'Multi-Fidelity Robust Optimisation using Subset Simulation', *Applied Mathematical Modelling*, Vol. 100, pp. 92-106.

[129] Gabrel, V., Murat, C. and Thiele, A. (2014) 'Recent advances in robust optimization: An overview', *European Journal of Operational Research*, Vol. 235, No. 3, pp. 471–483.

[130] Wang, L., and Kodiyalam, S., (2002) 'An efficient method for probabilistic and robust design with non-normal distributions', *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Denver, Colorado, 22-25 April, Paper No. AIAA-2002-1754.

[131] Ryan, K.M., (2014) *Robust multi-objective optimization of hypersonic vehicles under asymmetric roughness induced boundary-layer transition.* PhD Thesis, University of Maryland.

[132] Marzat, J., Walter, E. and Piet-Lahanier, H. (2013) ‘Worst-case global optimization of black-box functions through Kriging and relaxation’, *Journal of Global Optimization*, Vol. 55, No. 4, pp. 707-727.

[133] ur Rehman, S., Langelaar, M. and van Keulen, F. (2014) ‘Efficient Kriging-based robust optimization of unconstrained problems’ *Journal of Computational Science*, Vol. 5, No. 6, pp. 872-881.

[134] M, Kennedy, A. O’Hagan, (2001) ‘Bayesian Calibration of Computer Models’, *J. Royal Stat. Soc.* Vol. 63, No. 3, pp. 425-464.

[135] Oakley, J. (2002) ‘Eliciting Gaussian Process Priors for Complex Computer Codes’, *J. Royal Stat. Soc.* Vol. 51, No. 1, pp. 81-97.

[136] Xu, Q, Wehrle, E, and Baier, H. (2012) ‘Adaptive surrogate-based design optimization with expected improvement used as infill criterion’, *Optimization*, Vol. 61, No. 6, pp. 661-684.

[137] Arendt, P.D., Apley, D.W. and Chen, W. (2013) ‘Objective-Oriented Sequential Sampling for Simulation Based Robust Design Considering Multiple Sources of Uncertainty’. *Journal of Mechanical Design* Vol. 135 No. 5.

[138] Ling, Y. et al. (2018) ‘An intelligent sampling framework for multi-objective optimization in high dimensional design space’, *AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, doi: 10.2514/6.2018-0912

[139] Song, C., Yang, X. and Song, W. (2018) ‘Multi-infill strategy for kriging models used in variable fidelity optimization’, *Chinese Journal of Aeronautics*. Vol. 31, No. 3, pp. 448–456.

[140] Ghosh, S. et al. (2019) ‘A strategy for adaptive sampling of multi-fidelity Gaussian processes to reduce predictive uncertainty’, *Proceedings of the ASME Design Engineering Technical Conference*, doi: 10.1115/DETC2019-98418.

[141] Zhong-Hua, H., and Gortz, S. (2012) ‘Hierarchical kriging model for variable-fidelity surrogate modeling’, *AIAA Journal*, Vol. 50, No. 9, pp. 1885-1896.

[142] Toal, D.J.J., and Keane, A.J. (2011) ‘Efficient multipoint aerodynamic design optimization via cokriging’, *Journal of Aircraft*, Vol. 48, No. 5, pp. 1685–1695.

[143] Bailly, J., and Bailly, D. (2019) 'Multifidelity Aerodynamic Optimization of a Helicopter Rotor Blade', *AIAA Journal*, Vol. 57, No. 8, pp. 3132–3144.

[144] Ryan, K. M. et al. (2018) 'A Gaussian process modeling approach for fast robust design with uncertain inputs', *Proceedings of the ASME Turbo Expo*, doi: 10.1115/GT201877007.

[145] Li, Z., Wang, X., Ruan, S., Li, Z., Shen, C. and Zeng, Y., (2018). 'A modified hypervolume based expected improvement for multi-objective efficient global optimization method', *Structural and Multidisciplinary Optimization*, Vol. 58, No. 5, pp. 1961-1979.

[146] Zhan, D., Qian, J. and Cheng, Y. (2016) 'Pseudo expected improvement criterion for parallel EGO algorithm', *Journal of Global Optimization*, Vol. 68, No. 3, pp. 641-662.

[147] Zlatinov, M. and Laskowski, G. (2015) 'Hybrid Large-Eddy Simulation Optimization of a Fundamental Turbine Blade Turbulated Cooling Passage', *Journal of Propulsion and Power*, Vol. 31, No. 5.