Probabilistic analysis for computational mechanics with applications in Civil Engineering

David K. E. Green

A thesis in fulfilment of the requirements for the degree of Doctor of Philosophy

UNSW
AUSTRALIA

School of Civil and Environmental Engineering
Faculty of Engineering
University of New South Wales

February 2018

THE UNIVERSITY OF NEW SOUTH WALES
Thesis/Dissertation Sheet

Surname or Family name: Green

First name: David Other name/s: Kristopher Edmund

Abbreviation for degree as given in the University calendar: PhD

School: Civil and Environmental Engineering Faculty: Engineering

Title: Probabilistic analysis for computational mechanics with applications in Civil Engineering

Abstract

This thesis explores Uncertainty Quantification for probabilistic models of physical systems. In particular, the thesis works towards a long term goal of automating numerical methods by combining traditional techniques with perspectives from Machine Learning. Uncertainty Quantification refers to the use of probabilistic models for estimating the potential variability of unknown quantities that may be present. For engineering, the types of models of interest are those that estimate the behaviour of future states of the world by simulation. In particular, as these are relevant to the types of problems arising in Civil Engineering, Partial Differential Equations with probabilistic inputs are analysed. Uncertainty Quantification can be used to assess the risk associated with proposed designs and existing structures rigorously by explicitly calculating the variability of unknown quantities. This thesis begins by detailing how Uncertainty Quantification can be incorporated into a decision making framework by reference to Game Theory and Bayesian probability. Spatially distributed data is often encountered in Civil Engineering. Probabilistic models of spatial variability, random fields, are discussed extensively. Simulation and modelling techniques for random fields are presented and analysed to facilitate later developments in the thesis. Rare event reliability analysis is discussed in detail. Civil Engineering projects frequently have a ``low probability, high consequence" risk profile that presents unique computational challenges. Numerical methods for rare event probabilistic analysis based on Markov Chain Monte Carlo are demonstrated. The computational challenges associated with probabilistic computational mechanics are significant. If suitable self-improving numerical methods could be developed, then these computational challenges would be lessened. The thesis introduces an Artificial Neural Network surrogate model method to improve the efficiency of sampling based Uncertainty Quantification. Following this, the structure of probabilistic computational mechanics problems are analysed from a Bayesian perspective. From this interpretation of the solution of Partial Differential Equations, an adaptive Element Free Galerkin method is derived. The combination of Machine Learning, Bayesian probability and Partial Differential Equations presented indicates directions for future research in automated probabilistic numerical techniques.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

COPYRIGHT STATEMENT

AUTHENTICITY STATEMENT

ORIGINALITY STATEMENT

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed .....

Date .....

Abstract

This thesis explores Uncertainty Quantification for probabilistic models of physical systems. In particular, the thesis works towards a long term goal of automating numerical methods by combining traditional techniques with perspectives from Machine Learning. Uncertainty Quantification refers to the use of probabilistic models for estimating the potential variability of unknown quantities that may be present. For engineering, the types of models of interest are those that estimate the behaviour of future states of the world by simulation. In particular, as these are relevant to the types of problems arising in Civil Engineering, Partial Differential Equations with probabilistic inputs are analysed. Uncertainty Quantification can be used to assess the risk associated with proposed designs and existing structures rigorously by explicitly calculating the variability of unknown quantities. This thesis begins by detailing how Uncertainty Quantification can be incorporated into a decision making framework by reference to Game Theory and Bayesian probability. Spatially distributed data is often encountered in Civil Engineering. Probabilistic models of spatial variability, random fields, are discussed extensively. Simulation and modelling techniques for random fields are presented and analysed to facilitate later developments in the thesis. Rare event reliability analysis is discussed in detail. Civil Engineering projects frequently have a "low probability, high consequence" risk profile that presents unique computational challenges. Numerical methods for rare event probabilistic analysis based on Markov Chain Monte Carlo are demonstrated. The computational challenges associated with probabilistic computational mechanics are significant. If suitable self-improving numerical methods could be developed, then these computational challenges would be lessened. The thesis introduces an Artificial Neural Network surrogate model method to improve the efficiency of sampling based Uncertainty Quantification. Following this, the structure of probabilistic computational mechanics problems are analysed from a Bayesian perspective. From this interpretation of the solution of Partial Differential Equations, an adaptive Element Free Galerkin method is derived. The combination of Machine Learning, Bayesian probability and Partial Differential Equations presented indicates directions for future research in automated probabilistic numerical techniques.

Acknowledgements

I would like to include a brief statement of thanks to those that have contributed to this thesis.

First, the University of New South Wales and the Australian Government who have supported me with a scholarship.

A number of academics have contributed to this thesis. My supervisor Kurt Douglas for a few years now worth of guidance. My co-supervisor Wei Gao for suggesting I take some particularly useful course work. Also, Garry Mostyn for providing some of the original direction for what is now my line of work back when I started my undergraduate thesis many years ago. Of course, in the interest of brevity many go unmentioned here.

Finally, to my wife Brankica for putting up with yet another thesis.

Contents

List of Figures

List of Tables

Chapter 1

Overview

Contents

1.1 Introduction

This thesis presents a number of contributions in the area of Uncertainty Quantification for simulations of physical systems. Uncertainty Quantification, in the context of this thesis, uses probability theory to describe the variability of the inputs and outputs to computational methods. This thesis will be primarily concerned with Uncertainty Quantification of computational mechanical

models, that is, the variability associated with numerical simulations of physical phenomena. In particular, this thesis introduces extensions to existing Uncertainty Quantification methods before going on to present techniques for the synthesis of Uncertainty Quantification, Bayesian probability theory and recent developments in Machine Learning.

Uncertainty Quantification for computational mechanical problems involves the estimation of probability densities and expectation values for unknown quantities. Typically, this involves estimating a probability distribution over the inputs to a given simulation algorithm and then using numerical methods to estimate a probability distribution over the outputs of the simulation algorithm. The purpose of Uncertainty Quantification in this context is to facilitate decision making processes by enabling rigorous estimates of the future behaviour predicted by a chosen model of some physical process. Primarily, the physical systems of interest in this thesis are continuum mechanical problems modelled by Partial Differential Equations. The techniques detailed are, however, applicable to a wider range of methods for simulating physical systems. Uncertainty Quantification can be extended to other simulation techniques not based computational mechanics, but this is not the focus of this thesis. The particular applications presented are related primarily to Civil Engineering but, because the focus is on PDE modelling rather than particular systems, the techniques presented are applicable to virtually all fields that make use of PDE simulation for design and more general decision making.

In Civil Engineering, risk analysis for decision making regarding the design of large scale construction and infrastructure projects is crucial for both safety and efficiency. As the potential consequences of poor design in Civil Engineering may be catastrophic, focussing Uncertainty Quantification development in this area will hopefully have a large impact by enabling safer and more cost-effective infrastructure in the future. As detailed within this thesis, decision making that utilises Uncertainty Quantification can be more rigorously understood in terms of Game Theory. From a Game Theoretic perspective, physical system Uncertainty Quantification for decision making can be seen to be one aspect of a more general framework that also encapsulates risk, model selection and utility theory. The general Game Theoretic framework, in conjunction with probability theory, renders clear how the developments in this thesis can be used as a part of a general decision making process.

Simulations of physical behaviour are computationally demanding. Uncertainty Quantification introduces additional computational challenges and necessitates the use of numerical methods capable of dealing with these challenges. These challenges place Uncertainty Quantification at the intersection of physics, probability and information theory. This thesis details both theoretical and practical aspects of simulation based Uncertainty Quantification for physical systems. This involves simulation of the probabilistic inputs to a computational physics algorithm, computations using these algorithms and finally probability density and expectation value estimation over the output space for the computational model. For physical systems, the spatial character of the models is an important factor controlling the structure of the probabilistic models used. To generate and model the necessary input space probability distributions, this thesis details the random field theory (spatially autocorrelated probability distributions) required for this purpose.

The thesis then documents several approaches to numerical methods for Uncertainty Quantification over several Chapters. The thesis details sampling (Monte Carlo) based methods for estimating output probability distributions for physical systems. In particular, rare event reliability probability estimation using Subset Simulation and Markov Chain Monte Carlo for nonlinear continuum mechanical problems is detailed. Next, the thesis introduces techniques based on Machine Learning for estimating the output space probability distributions from computational models. These techniques leverage the self-improvement capabilities of Deep Artificial Neural Networks for adaptive computation of output response distributions. The thesis then returns to theoretical developments presented in earlier Chapters and describes a Bayesian interpretation of Uncertainty Quantification for physical systems. These theoretical developments suggest avenues for future research based on model selection uncertainty. Using the Bayesian perspective, the thesis adopts a probabilistic approach to numerical methods to derive an adaptive basis Element Free Galerkin method for solving PDEs based on the Expectation-Maximisation algorithm. This suggests that Variational Bayesian methods could be leveraged to derive improved algorithms in the future. The applications of Machine Learning to Uncertainty Quantification presented in this thesis, along with the Game Theoretic and Bayesian interpretation of Uncertainty Quantification for decision making, introduces the beginnings of a unified approach to decision making, risk, model selections and simulation. This general frame-

work will hopefully lead to deeper integration of Uncertainty Quantification into a variety of fields in the future.

1.2 Chapter summary

In this thesis an attempt will be made at advancing techniques for Uncertainty Quantification by making use of ideas from a range of different fields. The review of existing work is interspersed throughout the thesis. Efforts have been made, however, to ensure that surveys of existing literature are positioned near to the start of the relevant chapters.

The roadmap of the text is as follows:

- Chapter 1 introduces the Uncertainty Quantification problem from an intuitive perspective and presents this Chapter summary. Further, a number of mathematical preliminaries are presented in the Chapter 1 Appendix in preparation for the developments in the rest of the thesis.
- Chapter 2 begins by describing the necessary background in probability theory required for the remainder of the thesis. Using these developments, a rigorous interpretation of Uncertainty Quantification and how it relates to techniques for decision making in the face of uncertainty (with reference to Game Theory and Markov Decision Processes) is discussed. Following these discussions, the structure of methods suitable for Uncertainty Quantification is detailed. A brief review of existing methods for Uncertainty Quantification is also included.
- Chapter 3 introduces Random Fields as these can be used to model the type of spatially correlated data that arises in Civil Engineering problems. Theoretical background in random field theory is presented. Following this, a number of example simulation of Gaussian and non-Gaussian random fields are presented. An analysis of the computational complexity and numerical stability of random field sampling for Monte Carlo simulation of physical systems is also presented.
- Chapter 4 explores rare event reliability estimation, a difficult problem that arises in Civil Engineering, via nonlinear finite element analysis. Subset Simulation is applied to probability of failure estimation in non-

linear elasto-plastic finite element problems. The efficacy of different Markov Chain Monte Carlo methods for Subset Simulation are compared and contrasted. Further, a derivation of confidence intervals for the estimated relative error for Subset Simulation is presented. This new technique allows for vastly improved efficiency in the computation of error estimates for Subset Simulation.

- Chapter 5 demonstrates that Artificial Neural Networks can be used as effective regression surrogate models for Partial Differential Equation (PDE) Uncertainty Quantification. When the inputs to the equation are probabilistic, estimation of the output distribution is a computationally intensive. Calculation of the solution space distribution for a given input distribution requires evaluation of numerical PDE solutions many times. Surrogate models cache known solution values and attempt to generalise from known points to unknown values and can be used to estimate the output distribution. Feedforward Neural Networks can be trained to predict the solution of boundary value problems and fixed time initial value problems. Recurrent Neural Networks can be used for initial value problem time sequence prediction. Three numerical test case problems applying these techniques are presented.

- Chapter 6 extends the Bayesian interpretation of probabilistic numerical analysis by uncovering the connection between traditional approaches to Uncertainty Quantification and the Bayesian viewpoint. In particular, it is demonstrated that traditional approaches to Uncertainty Quantification are in fact a type of implicit Importance Sampling that uses Maximum Likelihood Estimates to generate sampling locations. The Bayesian approach to Uncertainty Quantification replaces minimum residual solutions with a probability distribution defined by the residual function of an equation to be solved. By exploring the link between the two paradigms, both the traditional forward and inverse approaches can be fully incorporated into the Bayesian viewpoint.

- Chapter 7 introduces a Bayesian form of Element Free Galerkin for the solution of PDE. By adopting a probabilistic perspective, it is possible to derive an adaptive basis PDE solver that does not rely on expanding the order of a function series. Instead, a Bayesian model likelihood is used to derive an Expectation-Maximisation type gradient descent algorithm for

iteratively improving basis functions. The basis functions are encoded in an Artificial Neural Network. Automatic Differentiation is leveraged to compute the necessary derivatives. The convergence of this method is demonstrated on a test case problem.

- Chapter 8 concludes the thesis by reflecting on the content of earlier Chapters and suggesting directions for future work.

1.3 Published contributions

The specific research contributions of this thesis published at the time of writing are as follows:

- Techniques suitable for simulating random fields in the presence of numerical stability problems, as well as an analysis of the computational complexity and numerical stability of these simulation methods. See Chapter 3. Published in [158].
- Demonstrated that Subset Simulation can be applied to rare event Uncertainty Quantification for nonlinear FEM problems. Further, derived confidence interval bounds that can be computed much more efficiently than was previously possible. See Chapter 4. Published in [157, 156].

1.4 Mathematical preliminaries and the structure of physical theories

To understand models of physical systems and how they can be analysed by Uncertainty Quantification methods, a number of mathematical preliminaries are detailed in the Chapter 1 Appendix. This background material is used to describe the structure of mathematical models of physical systems used in practice with particular reference to Partial Differential Equations (PDE). Knowledge of the material in the Chapter Appendix (primarily regarding Functional Analysis) will be assumed through the remainder of the thesis.

As this thesis is concerned with the simulation of physical systems, it is necessary to understand the mathematical anatomy of how these systems are modelled. Useful texts in this regard are [125, 277]. In modern physics, models of physical systems rely primarily on the conservation of various quantities. Conversation laws are directly related to symmetries of action functionals (Lagrangians). This is true of Quantum Field Theory, General Relativity and Classical Mechanics. As the focus within is on Civil Engineering problems, it is sufficient to consider only Classical Mechanics. Without the structure of these symmetries, there is nothing to restrict the manner in which future states may evolve as a result of earlier states. PDEs are the primary tool used to model the evolution of these states. Continuum mechanics [149, 250] is particularly relevant from a Civil Engineering perspective. Further relevant texts on continuum mechanics include [202, 62, 349, 27, 248, 14] and the references contained within [367].

Civil Engineering is a broad field and it is not possible to include direct numerical examples from all areas in a single thesis which is focussed on probabilistic methods. Instead, physical examples are mixed with the analysis of abstract probabilistic PDE models. The Uncertainty Quantification techniques explored in this thesis can (and are) utilised in all areas of Civil Engineering. Examples of probabilistic PDE models are found in structural engineering [141], geotechnical problems [118], groundwater hydrology [361, 362], contaminant transport [135] and many others [356]. The results presented in this thesis, which are demonstrated essentially independent of the type of PDE chosen, will be applicable across all of these areas. The interested reader will

hopefully be able to adapt the techniques presented within to their specific problem.

The theoretical developments in this thesis can be extended to non-PDE models used in Civil Engineering (such as Discrete Element Models [306]). However, a choice was made to restrict the numerical demonstrations to PDE as they are both fundamental to Classical Mechanics and sufficiently complex so as to not render the demonstrations trivial.

As well as those given in the following Sections of this discussion on preliminary definitions, background mathematical references relevant to the formalisms used in this thesis include:

- Logic, set theory and category theory [274, 247]
- Functional analysis [309, 221]
- Measure and Probability Theory [168]
- Topology, algebraic topology, differential topology [274, 172, 273, 163]
- Linear algebra [148, 191]
- Riemannian geometry and vector calculus [251, 236]
- Exterior algebra, exterior calculus [125, 86]

Important references usefully related to the analysis of PDEs for mechanics problems are given in [82, 389, 222] and the extensive reference lists in the volumes [331, 332]. For probabilistic modelling, relevant reference works with a Civil Engineering focus include [106, 171, 295].

Chapter 1 Appendix: Mathematical preliminaries

This Appendix describes key aspects of mathematics, in particular Functional Analysis. That is required to fully appreciate several aspects of the later Chapters of this thesis. Mathematical notation used throughout the thesis is also defined in this Appendix.

After setting out various terms from set theory and logic, progressively less abstract mathematical structures are defined. Abstract algebraic structures (Fields and Vector Spaces) are defined. Topological spaces, which define spaces

in terms of features not related to the distances between points within the space, are introduced. Metric spaces, which introduce notions of distance on sets, are defined. Norms, normed vector spaces and Hilbert spaces are discussed. Each of these structures are of crucial importance for understanding the solution of PDEs and the spectral decompositions used to simulate spatially autocorrelated probabilistic structures. The reader familiar with this material should feel free to skip forward to Chapter 2.

1.5.1 Sets theory and notation

At the fundamental level, mathematics requires notions from predicate and propositional logic before defining notions such as sets [379]. Assuming standard logic, this Section gives some notation and definitions that will be in use throughout the rest of the thesis. A useful overview is also provided in [285]. The definitions of logical relations are described in §2 of [108]. The following logical symbols from §1 of [285] are used in the text:

- Logical implication: $\Rightarrow$
- Logical equivalence: $\Leftrightarrow$

Axiomatic set theory is detailed fully in [226, 363]. Following §1 of [108], the definition of a *set* is taken as the intuitive definition that a set is a collection of elements. The notation here is adopted from §1 of [274].

If an *element*, a, is a member of a set, A, this will be denoted:

$$a \in A$$

If a does not belong in A then this will be denoted:

$$a \notin A$$

If two symbols for elements a and b of a set A refer to the same element, this will be denoted $a = b$. $A = B$ means that the symbols A and B refer to the same sets. Symbols enclosed by braces refers to a collection of elements of a set, for example for a set A composed of elements a, b and c can be written $A = \{a, b, c\}$. Similarly, $a \neq b$ and $A \neq B$ means that the symbols do not refer to the same element of set respectively. Further, the *set builder notation*

is used to specify sets by indicating properties that elements in the set must follows. Let A denote a set and $P(a)$ be a property that elements of A may or may not satisfy. Then a new set, B is defined by set-builder notation as the set of all elements of A such that $P(a)$ is true. This is denoted by:

$$B = \{a \in A | P(a)\} \tag{1.1}$$

or the equivalent form $B = \{a \in A : P(a)\}$ which uses : in place of |.

From [108], if a set B is a *subset* of A if every element of B is also an element of A. This is denoted $B \subseteq A$. If B is a subset of A that is also different from A (there exist elements of A which are not in B), this will be denoted $B \subset A$. In that case, B is a *proper subset* of A.

From [274], the *union* of sets A and B refers to the new set:

$$A \cup B = \{x | x \in A \text{ or } x \in B\} \tag{1.2}$$

The *intersection* of sets A and B refers to the new set:

$$A \cap B = \{x | x \in A \text{ and } x \in B\} \tag{1.3}$$

If there are no elements in $A \cap B$, the intersection is denoted by the *empty set*, $\emptyset$ (the set with no elements).

From [108], there are several formal definitions of an *ordered pair*. However, the most commonly adopted definition in modern mathematics is that an ordered pair of elements a and b in A is a a set of the form $\{\{a\}, \{a, b\}\}$. The intuitive definition of an ordered pair is suitable for this thesis. An ordered pair of elements $a, b \in A$ is denoted by (a, b) and means that the elements a and b are taken together in such a way that they are the first and second elements of the pair respectively. Two ordered pairs, (a, b) and (c, d), are equal if and only if $a = c$ and $b = d$. A *tuple* or n-tuple is an ordered collection of n elements analogous to an ordered pair, for example a 3-tuple of elements a, b and c would be denoted (a, b, c). Equality between tuples is defined pairwise between ordered elements of the tuples, that is $(a_1, a_2, \cdots, a_n) = (b_1, b_2, \cdots, b_n)$ if and only if $a_1 = b_1, \cdots, a_n = b_n$. There are several formal definitions of a tuple. When considering sets, tuples can be constructed from nested ordered pairs [108]. The elements of an ordered tuple may also be referred to as items.

The position within an ordered pair is referred to as the n-th *coordinate*. For example, in the tuple (a, b, c), the elements a, b and c are the items in the first, second and third coordinates respectively.

The *Cartesian product* will be the first formal definition specified in this thesis. From §1.2 of [108]:

Definition 1.5.1. *Cartesian product:* Given sets A and B, the *Cartesian product* of A and B is denoted by $A \times B$ and is the set of all ordered pairs (a, b) such that:

$$A \times B = \{(a, b) | a \in A \text{ and } b \in B\}$$

△

From §1.3 in [108] a *relation* is defined as:

Definition 1.5.2. *Relation:* Let A and B be sets. A *relation* R from A to B is a subset of $A \times B$ such that, for $(a, b) \in A \times B$, $a\ R\ b$ is true if $(a, b) \in R$. The set A is termed the *domain* of R and B is the *co-domain* of R. Denote $(a, b) \notin R$ by $a \not R\ b$. △

Following §2 [274], to define a *function* it is useful to first define a *rule of assignment*. A rule of assignment is a subset $R \in A \times B$ having the property that each element $a \in A$ appears as the first coordinate of at most one ordered pair in R. Then the *image set* of R is the subset of D consisting of all second coordinates of R:

$$\text{domain}(R) = \{a | \text{ there exists } b \in B \text{ such that } (a, b) \in R\}$$
$$\text{image}(R) = \{b | \text{ there exists } a \in A \text{ such that } (a, b) \in R\}$$

Then, a function is defined as:

Definition 1.5.3. *Function:* A *function* f is a rule of assignment, R, together with a set B that contains the image set of R. The domain of the rule R is also the domain of the function f. The image set of R is also the image set of f. The set B is the *range* of f. A function is denoted:

$$f : A \longrightarrow B$$

If $a \in A$, then $f(a)$ denotes the element of B that f assigns to a. △

Let $f^{-1} : B \to A$ denote the *inverse* of $f : A \to B$ (note that this function does not always exist, see §2 [274]). Let A_0 be a subset of A. Then $f(A_0)$

denotes the image of A_0 under f. Let B_0 be a subset of B. Then $f^{-1}(B_0)$ is the *preimage* of B_0 under f.

Other useful terminology for logical quantifiers, following §5 of [127] used in this thesis are the *universal quantifier* and the *existential quantifier*. The universal quantifier, denoted by $\forall$ (read as "for all"), can be taken to be short hand for the statements "for each", "for any" or "for all". For example, given a set A, $\forall a \in A$ would refer to all elements in A. Next, let : be short hand for the statement "for each". The existential quantifier, denoted by $\exists$ can be taken as shorthand for the statement "there exists". For example $\exists a \in A : a \in B$ says that there exists an element a in the set A such that the element a is B.

Further, other notation used in this thesis is as follows:

- $:=$ - definition symbol.
- $\sum_{i=1}^{n} x_i$ - sum of elements x_i indexed by i.
- $\prod_{i=1}^{n} x_i$ - product of elements x_i indexed by i.
- $\{x_i\}_{i=1}^{n}$ - the sequence of elements. $x_1, x_2, \cdots, x_n$ indexed by i.
- $\mathbb{N}$ - the set of integers.
- iff - shorthand for "if and only if".
- $\triangle$ - end of definition symbol.
- $\square$ - end of proof symbol.

1.5.2 Towards inner product spaces via vector and normed spaces

With basic notation in place, further progress requires the definition of a *field* (the most important example of which for the purposes of this thesis will be the real numbers, $\mathbb{R}$) which is a set with two operations defined, from §7 of [74] and Definition Definition 4.1.1 in [28], as follows:

Definition 1.5.4. *Field:* A *field* is a set, A, with two binary operations (operations that map two elements $a, b \in A$ to $c \in A$). The *addition* operation is denoted $a + b$. The *multiplication* operation is denoted $a \cdot b$. The operations satisfy the following properties (field axioms) for all $a, b, c \in A$:

- Associativity of addition and multiplication: $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
- Commutativity of addition and multiplication: $a+b = b+a$ and $a \cdot b = b \cdot a$.
- Additive and multiplicative identity: there exist two different elements, 0 and 1 in A such that $a + 0 = a$ and $a \cdot 1 = a$.
- Additive inverses: for every $a \in A$, $\exists b \in A$, called the additive inverse of a, denoted by $b = -a$, such that $a + (-a) = 0$.
- Multiplicative inverses: for every $a \neq 0 \in A$, $\exists b \in A$ called the multiplicative inverse of a, denoted by $1/a$, such that $a \cdot 1/a = 1$.
- Distributivity of multiplication over addition: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

△

As well as the real numbers, the complex numbers, $\mathbb{C}$ are also a field. Although not used within, a number of concepts described in this Section can be formulated in terms of other entities from abstract algebra, such as groups and rings. See [74, 28] for overviews of abstract algebra.

On the path towards geometry, the starting point will be the definition of a *topological space* from §2 of [274]:

Definition 1.5.5. *Topological space:* A *topology* on a set X is a collection $\mathcal{T}$ of subsets of X having the following properties:

- $\emptyset$ and X are in $\mathcal{T}$.
- The union of the elements of any subcollection of $\mathcal{T}$ is in $\mathcal{T}$.
- The intersection of the elements of any finite subcollection of $\mathcal{T}$ is in $\mathcal{T}$.

A set X for which a topology $\mathcal{T}$ has been specified is called a *topological space.* △

Roughly, topology defines the structure of mathematical spaces in terms of features not related to the distances between points within the space. The introduction of distance into a space introduces further rigidity in such a way that the usual features of mathematical spaces familiar to those solving PDEs for physical systems emerge. Following §1 of [221] and [309], the next step towards geometry is to introduce distance functions and metric spaces (from Definition 1.1-1 of [221]):

Definition 1.5.6. *Metric space and metric:* A *metric space* is a pair (X, d) where X is a set and d is a *metric on* X (also known as a *distance function*), which is a function

$$d : X \times X \to \mathbb{R} \tag{1.4}$$

such that for all $x, y, z \in X$ the following is true:

- d is real-valued, finite and nonnegative
- $d(x, y) = 0$ iff $x = y$
- $d(x, y) = d(y, x)$
- The *triangle inequality* holds, that is, $d(x, y) \leq d(x, z) + d(z, y)$

△

Note that metric spaces are a type of topological space (see §1.3 in [221]).

Introducing further structure requires notions of *vector spaces* and *normed spaces*. Vector spaces introduce operations between pairs of elements within a space (specifically vector addition and multiplication by scalars). In contrast with a metric (which assigns a distance between pairs of elements of sets) a *norm* defines the size of vectors within a space and can be used to define a metric. As such, all normed spaces are a type of metric space (§4.6 of [285]).

From Definition 2.1-1 in [221], a vector space is defined by:
Definition 1.5.7. *Vector space:* A *vector space over a field* K (whose elements are called scalars) is a nonempty set X of elements $x, y, z, \cdots \in X$ (called *vectors*) together with two algebraic operations called *vector addition* and *multiplication of vectors by scalars* defined by:

- Vector addition is a map $X \times X \to X$ that associates to each ordered pair of vectors (x, y) a vector $x + y$ (the sum of x and y) such that the following holds:
  - Vector addition is commutative: $x + y = y + x$
  - Vector addition is associative: $(x + y) + z = x + (y + z)$

  Further there exists is a vector 0, the *zero vector* such that
  - $x + 0 = 0$

Also, for every vector x there exists a vector $-x$ such that the following holds:

- $x + 0 = 0$
- $x + (-x) = 0$

- Multiplication of vectors by scalars is a map $K \times X \to X$ that associates to each vector $x \in X$ and scalar $\alpha \in K$ a vector αx (the product of x and α) such that, for $x, y \in X$ and $\alpha, \beta \in K$ the following holds:
  - $\alpha(\beta x) = (\alpha\beta)x$
  - $1x = x$
  - $\alpha(x + y) = \alpha x + \beta y$
  - $(\alpha + \beta)x = \alpha x + \beta y$

△

Vector spaces are frequently discussed in relation to their basis vectors. From §2.1 of [221], as elements within a vector space can be combined, a *linear combination* of vectors $v_1, v_2, \cdots, x_n \in V$ is an expression of the form:

$$\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n \tag{1.5}$$

or scalars $\alpha_1, \alpha_2, \cdots, \alpha_n$. A subspace of V is a nonempty subset $W \subset V$ such that $\forall w_1, w_2 \in W$ and scalars $\alpha, \beta \in W$, $\alpha w_1 + \beta w_2 \in W$. For $W \subset X$, $W \neq \emptyset$, the set of all linear combinations is called the *span* of W and is denoted span(W). A set, W, of n vectors in a vector space, if the only set of scalars for which a linear combination is zero, that is:

$$\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n = 0 \tag{1.6}$$

is true only for $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$ then the vectors in the set W are linearly independent. If W is not linearly independent then W is linearly dependent. Linear dependence can be used to define the *dimension of a vector space*. From Definition 2.1-7 of [221]:

Definition 1.5.8. *Dimension of a vector space and basis vectors:* A vector space, V, is *finite dimensional* if there is a positive integer n such that V contains a linearly independent set of n vectors while any set of $n + 1$ vectors

is linearly dependent. Then n is the dimension of V, denoted $\dim(V)$. If X is not finite dimensional, it is infinite dimensional. If $\dim(V) = n$, a linearly dependent n-tuple of vectors of V is called a *basis* of V. $\triangle$

Given a basis, it is possible to represent any vector in a vector space V as a linear combination of the basis vectors. For some $v \in V$, the values of the scalars used to multiply the basis vectors are called the *components* of the vector v.

Real and complex vector spaces are taken to refer to vector spaces over the field of real numbers, $\mathbb{R}$, or complex numbers, $\mathbb{C}$, respectively. The typical example of a vector space used in engineering type applications, the space of N-dimensional real numbers $\mathbb{R}^N$, is often used with the intuitive Euclidean notion of distance. However, there is no direct link between the algebraic notion of a vector space as defined above and the definition of a metric space on which has a notion of distance is present. As outlined in §2 of [309] and §2.2 of [221], it is necessary to first introduce the concept of a *norm*. A norm gives, in a sense, a way to measure the size of elements of a vector space. Using a norm, a metric can be constructed. Via this route, the usual intuitive notions of geometry can be recovered. From Definition §2.2-1 of [221] and §1 [309], a normed space is defined as follows:

Definition 1.5.9. *Normed space:* A *normed space* X is a pair $(V, \|\circ\|)$ where V is a vector space with the norm $\|\circ\|$ defined on it. A *norm* on a real or complex vector space is a real-valued function on X whose value at $x \in X$ (the norm of x) is denoted $\|x\|$ with the following properties (for x, y vectors in V and α is a scalar in V):

- $\|x\| \geq 0$
- $\|x\| = 0$ iff $x = 0$
- $\|\alpha x\| \geq 0$
- The triangle inequality: $\|x + y\| \leq \|y\|$

A norm on X defines a metric d on X, the *metric induced by the norm*, given by:

$$d(x, y) = \|x - y\| \qquad \text{for } x, y \in X \tag{1.7}$$

$\triangle$

A Banach space is a particular type of normed space. Specifically a Banach space is a normed space that is *complete* in the metric defined by the norm. Norms and notions of completeness allow for concepts such as continuity to be defined rigorously. See [309, 217] for more details. From §1 in [309], a *Cauchy sequence* is defined as:

Definition 1.5.10. *Cauchy sequence:* Given a metric space, (X, d), a sequence of elements $\{x_n\} \in (X, d)$ is called a *Cauchy sequence* if $(\forall \epsilon > 0)(\exists N)$ such that $n, m \geq N$ (where n, m and N are integers) implies $d(x_n, x_m) < \epsilon$. $\triangle$

A sequence $\{x_n\} \in (X, d)$ is said to *converge* to $x \in X$ if $d(x, x_n)$ approaches zero as n approaches infinity (see [309, 217] for rigorous definitions). Further, all convergent sequences are Cauchy. A metric space in which all Cauchy sequences converge is called *complete*. Intuitively, completeness means that sequences of elements in X converge to another point in X such that, in a sense, there are no points "outside" of X.

1.5.3 Operators and functionals

Operators and operator theory provide a means to understand functions between vector (and other) spaces. From this point on, the terms operator, function and mapping will be used essentially interchangeably. From §2.6 in [221], a mapping between vector spaces is termed an *operator*. Different types of operators have different mathematical properties. Of particular importance are *linear operators*. From Definition 2.6-1 in [221]:

Definition 1.5.11. *Linear operator:* A *linear operator* T is an operator such that:

- The domain of T is a vector space and the range of T lies in a vector space over the same field.
- $\forall x, y \in \text{domain}(T)$ and scalars α:

$$T(x + y) = Tx + Ty$$
$$T(\alpha x) = \alpha Tx$$

Note that the *null space* of T is the set of all $x \in \text{domain}(T)$ such that $Tx = 0$. $\triangle$

The linearity of an operator is also often expressed by writing:

$$T(\alpha x + \beta y) = \alpha Tx + \beta Ty$$

Due to the centrality of the operator concept in essentially all of the relevant mathematical developments in this thesis, it is worth considering a few relevant examples. For example, given the vector space of all polynomials on the interval $[a, b] \in \mathbb{R}$, differentiation is the linear operator $Tx(t) = \frac{dx(t)}{dt}$. Similarly, given the space of all continuous functions on the space $[a, b] \in \mathbb{R}$, denoted $C[a, b]$ (see §1 [221]), integration can be seen to be a linear operator where $Tx(t) = \int_a^b x(s)ds$. Finally, standard operations from linear algebra such as matrix multiplication and vector products (for more details see [191]) are operators on finite dimensional vector spaces.

A *linear functional* refers to a linear operator that maps vectors in a vector space to the scalar field of the vector space. From §2.8-1 of [221], a linear functional is defined as:

Definition 1.5.12. *Linear functional:* A *linear functional* f is a linear operator with domain in a vector space, V, and range in the scalar field K such that:

$$f : \text{domain}(f) \longrightarrow K \tag{1.8}$$

where $K = \mathbb{R}$ is V is real and $\mathbb{C}$ if V is complex. △

Norms and definite integrals are important examples of linear functionals [221, 309]. Introducing linear functionals brings this Section closer to the goal of defining inner product spaces.

1.5.4 Dual spaces

With the definitions of vector spaces and linear functional in place, it is useful to consider the structure of linear functionals on vector spaces. From §2.1 of [125], consider a vector space V of dimension n with basis $e_1, e_2, \cdots, e_n$. Then $v \in V$ can be written:

$$v = \sum_{i=1}^{n} \alpha^i e_i \tag{1.9}$$

Note that here, subscripts and superscripts are indices and not powers. The use of raised and lowered indices to represent vectors and scalars in this way

is a notational convenience for later concepts in abstract algebra and geometry.

Foreshadowing later developments in this Section, in matrix notation (see [148], this can be written such that the scalars v^i (known as *components*) are the entries of a column vector a and the basis vectors e_i are the entries of a row vector, e so that $v = ea$ such that v is a one by one matrix.

Let α be a linear functional on the n dimensional real vector space V such that $\alpha : V \to \mathbb{R}$. Due to linearity, $\alpha(av + bw) = a\alpha(v) + b\alpha(w)$ for $a, b \in \mathbb{R}$ and $v, w \in V$. By induction and the linearity of α it can be shown that, for any basis e:

$$\alpha\left(\sum_{i=1}^{n} v^i e_i\right) = \sum_{i=1}^{n} v^i \alpha\left(e_i\right) \tag{1.10}$$

Defining $a_i := \alpha(e_i)$, $\alpha\left(v\right) = \sum_{i=1}^{n} v^i \alpha_i$. Then, a selection of the scalars α_i defines a linear functional on V.

Following §2.1 of [125], the collection of all linear functionals on a real vector space are defined to be the *dual space*:

Definition 1.5.13. *Dual space:* The collection of all linear functionals α on a vector space V form a new vector space called the *dual space* to V denoted V^* under the operations:

$$\begin{aligned} (\alpha + \beta)(v) &:= \alpha(v) + \beta(v) \quad \alpha, \beta \in V^* \quad v \in V \\ (c\alpha)(v) &:= c\alpha(v) \quad c \in \mathbb{R} \end{aligned}$$

△

It is important to note that the dual space is not the same space as the original vector space. This is discussed in §2.1 of [125].

From §2.1 of [125], if a vector space V is n dimensional, then so is V^*. This can be seen by introducing the *dual basis*. The dual basis is a particular, convenient, selection of the vectors in V^*. First, define the Kroenecker delta:

$$\delta_j^i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Then, if $e_1, \cdots, e_n$ are elements of the basis e of V, then define the dual basis

$e^1, \cdots, e^n$ of V^* by setting $e^i(e_i) := \delta^i_j$. With this definition, the dual basis selects the components a vector in V as follows:

$$\begin{aligned} e^j(v) &= e^j(\sum_{i=1}^{n} v^i e_i) \\ &= \sum_{i=1}^{n} v^i e^j(e_i) \\ &= \sum_{i=1}^{n} v^i \delta^j_i \\ &= v^i \end{aligned}$$

That the dual basis spans V^* and is, as such, a basis of the dual space of V is shown in §2.1 of [125]. Then any linear functional $\alpha \in V^*$ can be written as a linear combination over the dual basis $\alpha = \sum_{i=1}^{n} \alpha_i e^j$ such that:

$$\begin{aligned} \alpha(v) &= \sum_{i=1}^{n} v^i \alpha(e_i) \\ &= \sum_{i=1}^{n} v^i \sum_{j=1}^{n} \alpha_j e^j(e_i) \\ &= \sum_{i=1}^{n} v^i \sum_{j=1}^{n} \alpha_j \delta^j_i \\ &= \sum_{i=1}^{n} v^i \alpha_i \end{aligned}$$

In terms of matrix linear algebra, the components of the linear functional, α are written as a row matrix with i-th entry given by α_i.

1.5.5 Inner product spaces

Having described linear functionals, inner product spaces can be introduced. Whereas linear functionals are elements of the dual space to a vector space V that map vectors to the scalar field of V, an inner product is a map from two vectors of V to the scalar field of V. This is a subtle difference but is significant mathematically.

Norms on vector spaces allow for the size of vectors in a space to be defined.

Inner products introduce additional structure on a vector space, allowing for the relative orientation of vectors to be defined in terms of their degree of orthogonality.

From §II.1 of [309], Definition 3.1-1 of [221] and §6.1 of [285] an *inner product space* is defined as:

Definition 1.5.14. *Inner product space:* A real (or complex) vector space V over the field K is called an *inner product space* if there is a real (or complex) valued function called an *inner product* defined by $\langle \cdot, \cdot \rangle : V \times V \to K$ that satisfies, for vectors $x, y, z \in V$ and scalar $\alpha \in K$, the following conditions:

- $\langle x, x \rangle \geq 0$ and $\langle x, x \rangle = 0$ if and only if $x = 0$
- $\langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle$
- $\langle x, \alpha y \rangle = \alpha \langle x, y \rangle$
- $\langle x, y \rangle = \overline{\langle y, x \rangle}$

where $\overline{\langle y, x \rangle}$ denotes complex conjugation. For $K = \mathbb{R}$, $\overline{\langle y, x \rangle} = \langle y, x \rangle$. △

An inner product on V defines a norm on V, for $x, y \in V$, by:

$$\|x\| = \sqrt{\langle x, x \rangle}$$

Further, an inner product defines a metric on V by:

$$d(x, y) = \|x - y\| = \sqrt{\langle x - y, x - y \rangle}$$

A *Hilbert Space* is a complete (in the metric given by the inner product) inner product space.

Importantly, the norm on an inner product space satisfies the parallelogram inequality:

$$\|x - y\|^2 + \|x - y\|^2 = 2\left(\|x\|^2 + \|y\|^2\right)$$

Using this fact, following from §3.1 in [221], it can be shown that not all normed spaces are inner product spaces.

Inner products allow for orthogonality of vectors to be defined, from Definition 3.1-2 in [221] and §II.1 in [309], by:

Definition 1.5.15. *Orthogonal vectors:* An element, v, of an inner product

space, V, is said to be *orthogonal* to another element $w \in V$ if

$$\langle v, w \rangle = 0 \tag{1.11}$$

A collection of vectors $\{x_i\} \in V$ is called an *orthonormal set* if $\langle x_i, x_j \rangle = \delta_{ij}$ where δ_{ij} denotes the Kronecker delta. $\triangle$

Orthogonality is of great help when considering basis vectors of inner product spaces. Given an orthonormal set of vectors $\{e_i\}_{i=1}^n \in V$ that is also a basis with $\text{span}(\{e_i\}_{i=1}^n) = V$ then the vector $v \in V$ can be written as:

$$v = \sum_{i=1}^{n} v^i e_i$$

From the properties of the inner product, the inner product of v with a fixed element e_j from the orthogonal basis set $\{e_i\}_{i=1}^n$ is:

$$\begin{aligned}
\langle v, e_j \rangle &= \langle \sum_{i=1}^{n} v^i e_i, e_j \rangle \\
&= \sum_{i=1}^{n} v^i \langle e_i, e_j \rangle \\
&= \sum_{i=1}^{n} v^i \delta_{ij} \\
&= v^j
\end{aligned}$$

As such, coefficients of the vector v in the orthogonal basis set $\{e_i\}_{i=1}^n$ can be written:

$$v = \sum_{i=1}^{n} \langle v, e_i \rangle e_i$$

From §2.1 in [125], the inner product of basis vectors need not be orthogonal. In this case, the coefficients $\langle e_i, e_j \rangle = g_{ij}$ are said to be the components of the *metric tensor*.

1.5.5.1 Important Hilbert Spaces - Euclidean space and $\mathcal{L}^2$ functions

Standard Euclidean space, $\mathbb{R}^n = \mathbb{R} \times \cdots \times \mathbb{R}^n$ with the standard vector dot product from linear algebra is a Hilbert space (see §3.1 of [221]). The vector dot product on Euclidean space is defined, given an orthonormal basis, $\{e_i\}_{i=1}^n$ spanning $\mathbb{R}^n$, as:

$$\begin{aligned}\langle x, y\rangle &= \langle \sum_{i=1}^{n} x^i e_i, \sum_{j=1}^{n} y^i e_i \rangle \\ &= \sum_{i=1}^{n} x^i y^i\end{aligned}$$

In many circumstances, the integrals of functions over spaces are linear functionals as the integrals map from vector spaces (of functions) to scalars. Convergent integrals (those that are less than infinity) have special properties. See [221] for details on the boundedness of operators. Among the so-called $\mathcal{L}^p$ spaces, the $\mathcal{L}^2$ space is the only Hilbert Space. All other $\mathcal{L}^p$ spaces are (or can be completed to become) Banach spaces (see §III in [309]). In Chapter 2, basic measure theory is outlined including the definitions required to more rigorously appreciate $\mathcal{L}^p$ spaces. At this stage, however, the reader can appeal to their intuition regarding integration of functions over $\mathbb{R}^n$. From §III in [309] and §29 in [217], the $\mathcal{L}^p$ space is defined as follows:

Definition 1.5.16. *$\mathcal{L}^p$ space:* Let (X, Σ, μ) be a measure space (see Definition 2.5.1) and $p > 1$. Then $\mathcal{L}^p(X, d\mu)$ is the set of measurable functions (see Definition 2.5.3) $f : X \to \mathbb{R}$ (or $\mathbb{C}$) which satisfy:

$$\|f\|_p := \left(\int_X |f(x)|^p \right)^{\frac{1}{p}} < \infty \tag{1.12}$$

where $|f(x)|^p$ denotes the absolute value of $f(x)$ to the p-th power. △

The set of functions $f \in \mathcal{L}^p(X, d\mu)$ form a vector space. Under the conditions discussed in [309], the functions $f \in \mathcal{L}^p(X, d\mu)$ are a Banach space.

In the complex case, $\mathcal{L}^2(X, d\mu)$ is a Hilbert space with inner product defined

for $f, g \in \mathcal{L}^2(X, d\mu)$ by:

$$\langle f, g \rangle = \int_X f(x)\overline{g(x)}d\mu(x)$$

where $\overline{g(x)}$ denotes the complex conjugate of $g(x)$. In the real valued case, $\overline{g(x)} = g(x)$. The Hilbert Space of functions in $\mathcal{L}^2$ are also known as *square-integrable functions*.

Chapter 2

Uncertainty Quantification for Computational Mechanics

Contents

Figures

Chapter 2 Overview

Key developments in Chapter 2 include:

- Section 2.2 contributes a discussion on how probabilistic modelling for engineering problems can be related to decision making under uncertainty frameworks. Particular reference is made to Game Theoretic concepts as well as notions such as the Value of Perfect Information.
- Section 2.3 presents background definitions from Uncertainty Quantification and discusses the need for and structure of probabilistic engineering models. Interpretations of probabilistic modelling needed for later developments in the thesis are detailed.
- The Chapter 2 Appendix presents further background material on Probability Theory and Bayes Theorem. In particular measure theory, Bayesian inference over time and Hidden Markov Models are discussed.

2.1 Introduction

This goal of this Chapter is to clarify the roles and aims of Uncertainty Quantification for physical systems modelling and to discuss how Uncertainty Quantification can be understood as a part of a decision making process more generally. To this end, this Chapter discusses Uncertainty Quantification from the perspective of Game Theory, that is, in terms of agents that seek to maximise some utility function over time. For example, the designers of an infrastructure project may seek to maximise their utility by reducing the risk associated

to the project by minimising the probability that their designs will fail. Uncertainty Quantification seeks to rigorously assess the types of probabilities needed to characterise risks of this sort numerically.

In particular, Uncertainty Quantification for physical models that aim to predict the future behaviour of physical systems is considered by considering 'decision making in the face of uncertainty' models. Along with the Chapter 2 Appendix, this Chapter outlines the required background material needed to define the Uncertainty Quantification. Uncertainty Quantification for physical systems is considered from the perspective of planning, which can be considered to be the use of past and present knowledge to predict expected future states. The computational resources required to undertake Uncertainty Quantification analyses are discussed in terms of the Value of Information concept. By considering numerical simulations in this way, their place within a complete decision making framework can be understood. With this theoretical structure in place, Uncertainty Quantification for physical system models is considered from two perspectives. First a top-down approach, which seeks to define the structure of a hypothetical optimal method for Uncertainty Quantification, is discussed. Next, an overview of existing methods for Uncertainty Quantification is presented. By considering the formal structure of Uncertainty Quantification, questions of how to interpret probabilistic simulation methodologies are addressed in preparation for the material presented in the subsequent Chapters of this thesis.

2.2 Making decisions in the face of uncertainty

The Section discusses uses of inference by considering decision making processes. The theory of how an agent can make decisions to achieve desired outcomes based on beliefs about the world is founded on probability and utility theory. Utility defines the relative desirability of a given outcome. Combining inference, discussed in the previous Section, with utility considerations leads to the problem of estimating what actions to take in order to maximise utility. Game Theory can be applied to encapsulate different forms of sequential decision making problems, that is, how an agent should act in order to optimally maximise utility in a manner that balances risks and rewards over time. Various specific formulations of sequential decision making

problems that are commonly used in practice include *Markov Decision Problems* (MDP) and *Partially Observable Markov Decision Problems* (POMDP). The brief overview of these topics in decision making theory presented here does not seek to present detailed exposition of all of these vast subject areas. Rather, this Section presents a sufficient set of terms from Game Theory such that Uncertainty Quantification (in particular in relation to modelling of physical systems for science and engineering) can be clearly understood. A more full treatment on Game Theory is presented in [369]. The presentation here will help to clarify the meaning of terms frequently encountered in engineering design, such as *acceptable risk* [95], and how these can be related to a more general cost-benefit type analysis framework.

Game Theory is a framework for understanding how an agent can estimate what actions to take in order to maximise some utility function over time, where the utility function assigns a level of desirability to different possible outcomes. These utility functions can be subjective (dependent on the preferences of the agent) and can be defined a variety of ways. When making decisions to maximise utility, the full state of the world may be unknown. In this case, because not all information is available to an agent before making a decision, there is a value that can be placed on information by analysing the potential utility gain from performing exploratory actions to minimise uncertainty. These exploratory actions could potentially include expending resources to make computations with models of the world to predict future world behaviours (as in planning) or performing actions in the real world with the intent to gather more information (for example, prospecting for mineral deposits before excavating a mine). The Value of Information is further described in this Section.

2.2.1 Utility Theory and Decision Making

For this thesis, philosophical debates regarding utility functions will be avoided. Instead, it will be assumed that some subjective function, the *utility function*, can be defined that represents the preferences of an agent. For a detailed discussion see §1 of [369]. Following §16 of [323], a utility function is a map $U : S \to \mathbb{R}$ from a set of states S to real numbers such that the desirability of states is represented by the utility of the state (with higher numbers representing higher desirability) . Utility functions can be derived by consid-

ering an ordering relation on the desirability of different states. Then, the goal of a utility maximising agent is to maximise the total utility over time. The *discounted sum of future reward* is a typical example of a time dependent utility function (equation (2.1) and §17.1.2 of [323]). In the time dependent case, an agent may move through a series of states in S, $(S_1, \cdots, S_t, \cdots)$, over time. At each time step an agent may take an action, a, based on a policy, $\pi(a|s) = P(a|s)$, which influences future states. A policy defines the probability to take an action a given a set of possible actions A in state $s \in S$. In the discounted sum of future reward case, an agent seeks to maximise, over all times t from $t = 0$ to $t = \infty$ the value of:

$$U(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma R(S_t)\right] \tag{2.1}$$

where $U(s)$ is the utility (or *value*) of the initial state $s \in S$ and γ is a discounting factor used to express the relative desirability of near-future states over far-future states. Under this model, the function $R(S_t)$ is the *reward function* which assigns immediate rewards upon entering a state. In this setting, the reward function is represented as a number in $\mathbb{R}$ that assigns higher values to more desirable rewards.

Reward and utility functions can be selected by agents subjectively or assigned to an agent. In the context of Uncertainty Quantification, utility may represent any number of outcome preference orderings. For example, utility may represent the desirability of the completion of an infrastructure problem versus the undesirability of a large scale failure. Reward functions may represent, for example, happiness, monetary rewards or (as a negative reward) painful states (such as fines incurred due to design failures). A deeper analysis of reward functions and utility theory is presented in [369]. Information itself can have a utility if it can be leveraged to obtain additional future rewards. This Value of Information is the key to understanding why Uncertainty Quantification can be effective when attempting to predict the behaviour of physical systems in order reach desired future outcomes (such as the completion of some engineering project).

If an agent can take actions that influence future states, then it is possible to consider a decision making process in which an agent attempts to maximise future rewards by appropriate action selection. In a typical sequential decision

making process, it is necessary to learn both the reward function (what rewards are received in what states) and the action policy (how to maximise these rewards over time). There are a large number of so-called environment or game models describing how states, actions and rewards influence each other. This is particularly relevant when considering different types of (Game Theoretic) games. For example, different models may consider whether the transitions between states are deterministic or probabilistic, if all states are observable or if certain states are hidden and so on. For further examples, see [369]. The *Markov Decision Process* (MDP) is the simplest relevant model and is described in this Section. Using the MDP as a reference point, the use of Uncertainty Quantification for estimating the behaviour of physical systems can be understood and discussed clearly.

After MDPs are introduced, this Section discusses reinforcement learning and planning. Planning, in particular, can be viewed as a technique to estimate desirable future states. From this discussion, Uncertainty Quantification can be related to planning for a decision making process. Models of physical systems can be understood as state transition models (in the sense of time-dependent Bayesian Inference) that attempt to model future states given present states based on previous observations of natural phenomena.

2.2.1.1 Markov Decision Process

The Markov Decision Process (MDP) is a useful framework for many problems that can be used to facilitate understanding of models for decision making. An MDP is a discrete time stochastic control problem defined as:

Definition 2.2.1. *Markov Decision Process:* A (discrete time) Markov Decision Process is a tuple (S, A, P, R, γ) where:

- S is a finite set of states.
- A_s is a finite set of actions available in state $s \in S$.
- $P_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that, after taking action a in state s at time t, the state at time $t+1$ will be $s' \in S$.
- $R_a(s, s')$ is the expected immediate reward received upon transitioning to state s' after taking action a in state s.
- γ is the future reward discount factor defined as in equation (2.1).

such that the utility of a state is given by equation (2.1). △

In words, an MDP models transitions between the various states of S due to taking actions $a \in A$. At each time, an agent occupies a single state and can take a single action. After taking action a at time t in state s, the agent is immediately transitioned to the state s' according to the probabilities defined by $P_a(s, s')$. After transitioning to state s', a reward $R_a(s', s)$ is given to the agent. These rewards may be probabilistic (as in so-called bandit problems [366]). The goal of an agent in an MDP is to maximise the rewards over time by selecting actions that will lead to states that provide high rewards. The *value function* is defined as the expected future rewards, $U(s)$, given an initial state s and is in this case equal to equation (2.1). Alternative value function formulations can be defined depending on the structure of the problem being analysed, as discussed extensively in [366]. The value function can be seen as an encoding of the solution of a dynamic programming problem. The *policy function* defines what action an agent will take in a given state, $\pi(s) = P(a|s)$. In the MDP case, it is possible to iteratively improve the value and policy functions separately and still converge on a solution. These procedures are known, respectively, as value iteration and policy iteration and is detailed in [366].

In the more general case, the state that an agent is in unobserved. In this case, decision making requires an observation model to be used to estimate the hidden true state. The Markov formulation of such a scenario is typically specified as the Partially Observable Markov Decision Process (POMDP) model (see §17 of [323]). Continuous time Markov dynamics can be modelled as stochastic partial differential equations [189]. The POMDP case reflects sequential decision making problems in the real world more closely than the fully observed case in that the entire universe is not observable to individuals at all times. Instead, inferences must be made about the current state an individual is in, as well as likely high utility future states. Uncertainty Quantification facilitates future state estimation by using simulations to predict the evolution of physical systems in time.

2.2.1.2 The Value of Information

Estimating future states based on present information (planning) requires the use of computational resources. In order for Uncertainty Quantification to be worthwhile, it is necessary that the opportunity cost of expending resources on gaining new information (rather than on something else) is balanced by the future rewards that are potentially available given that information. Value of Information analysis (or information value theory) addresses this problem (see §16.6 [323]). This is often modelled using the *Expected Value of Perfect Information*. A clear example is provided by mineral resource prospecting. Consider the problem that some valuable underground resource is unknown. Investigating the location of high resource yield areas requires expensive drilling. Building a mine to extract the resource is much more costly than exploratory drilling, but will be profitable is the mine is placed in a high yield area. The drilling information regarding the resource quantities has a potential value in that it allows for the mine to be constructed in the correct place.

Following §16.2 of [323], consider a random variable E_j for which exact evidence of the value of E_j, denoted e_j, can be obtained. Given the current evidence, e, the value of the best action, a', is defined by:

$$\mathbb{E}\left[U(a'|e)\right] = \max_{a \in A} \sum_{s'} P(s'|a, e)U(s') \tag{2.2}$$

In other words, the expected utility of the action a given the available evidence e is calculated from the maximum utility resultant state, s', reached after taking action $a' = \max_{a \in A}$. Assume that evidence, e_j, about the value of E_j is received upon reaching the new state s'. The best action, α_j, after obtaining new evidence, e_j, would then be given by:

$$\mathbb{E}\left[U(a_j|e, e_j)\right] = \max_{a \in A} \sum_{s'} P(s'|a, e, e_j)U(s') \tag{2.3}$$

As E_j is a random variable, it is necessary to consider the expected utility due reward gained by transitioning to the states s' and then collecting information upon arriving in the states s'. As the future evidence, e_{jk}, is only found during state transitions, the anticipated values of e_{jk} must be estimated using incomplete present knowledge, e. Then, the Value of Perfect Information can

be defined as:

$$VPI_e(E_j) = \left(\sum_k P(E_j = e_{jk}|e)\mathbb{E}\left[U(a_j|e, e_j)\right] \right) - \mathbb{E}\left[U(a'|e)\right] \tag{2.4}$$

Equation 2.4 says that the random variable E_j is valuable if it has an influence on future planning. If the expected action utility is unaltered by learning about E_j, then the term $\mathbb{E}\left[U(a'|e)\right]$ (that gives the expected utility taking expected optimal actions given present knowledge about E_j) will force $VPI_e(E_j) = 0$. This is because expected optimal course of action given the observed evidence, defined as the utility maximising path given new observations expressed as $(\sum_k P(E_j = e_{jk}|e)\mathbb{E}\left[U(a_j|e, e_j)\right])$, will not have a higher expected value than the action choices that would be made given the present information. If new plans, given new knowledge about E_j, have a higher utility then $VPI_e(E_j)$ will be greater than zero. Although on average the expected utility of an information gathering course of action may be higher than one which does not seek to gather new information, an actual trajectory over information gathering states may lead to lower utility. That is, expected utility returns model distributions over several states but only one realisation of a trajectory of states will actually be realised. Returning to the prospecting example given above, a drilling location may be selected on the basis that it is expected that a high concentration of the desired resource is located there. The future mine is anticipated to recover the costs of the initial drilling. Upon actually drilling in an expected likely high concentration location, it may be that the prior beliefs were incorrect and there are no resources to be extracted.

In relation to Uncertainty Quantification (the use of physical system simulations for planning expected future states of nature), the Value of Information can be used to further reason about effective engineering design methodologies. For instance, a data collection program may help to reduce uncertainty regarding material property strengths. As a Civil Engineering example, consider the construction of a building on soil foundations. If the soil properties are unknown, to avoid collapse it is necessary to design the foundation assuming that it is possible for very low strengths to be present. Thicker, stronger footings are more costly than simpler footings designed to withstand lower forces. If testing can confirm that low soil strengths are very unlikely, a simpler and cheaper footing can be constructed. Extracting and analysing the

strengths of soil samples from the foundation location will incur costs, but these may be offset by the cheaper design.

Returning to simulations of physical systems, costs will be incurred in terms of the time and energy spent in running the various solver algorithms required. If these algorithms can accurately predict future states (for example by predicting whether a bridge can withstand the loads expected) then the likelihood of an agent finding itself in some future state can be estimated. If future state likelihoods can be estimated, then the agents expected future utility can also be estimated. The evidence variable in this case is the information gained by running simulations, which will help to restrict the expected range of future states to those with that are more likely based on simulated knowledge. If, for example, running a simulation can identify that a bridge is very likely to collapse, then actions can be taken in the present (such as altering the bridge design) to avoid low utility states in the future. The computational costs of the simulations lower utility (relative to a hypothetical cost free simulation) and must be balanced against the accuracy of the simulations. Inaccurate simulations should be afforded a comparatively lower belief (higher expected variability) in the estimated outputs than a 'better' simulation. More complex simulations may require more effort to run, giving a more accurate prediction of future states but at higher cost. By considering the Value of Information it is clear that, for engineering and scientific design, it is ideal to use simulation methodologies that are as accurate as possible for the minimum computational effort. For this reason, the computational complexity of algorithms used for Uncertainty Quantification should be analysed.

2.2.2 Reinforcement Learning and Planning for Sequential Decision Problems

Solving a sequential decision problem typically refers to learning both the value function and the optimal policy required to reach maximum utility states. The full reinforcement learning problem involves learning the system dynamics (the reward, transition and observation models) as well as the value and policy functions using feedback from the environment. For these problems, utility maximisation is typically formulated as a dynamic programming problem that attempts to learn the value and policy functions that will optimise future rewards. In particular, Jacobi-Bellman based approaches are used to justify

the use of value functions as a method of working backwards from goal states to present states. These issues are discussed extensively from the perspective of reinforcement learning in [366].

Planning solves a simplification of the full reinforcement learning problem. Given a desired goal state and assumed known system dynamics, planning asks what actions should be taken to reach the desired goal [218]. Planning can be used to reduce the computational resources required to model the entire environment and to predict how actions influence future states. For example *partial policies* (see §4 of [218]), which specify actions for some (rather than in all) states, provide one such heuristic approach to reducing the burden of storing action choices for all possible states. As computational resources are always finite (see [124, 245]), when considering real world problems the planning capacity of an agent is always restricted. It is necessary to make partial estimates of future states. The combination of planning (estimating future environmental feedback) and reinforcement learning (received actual environmental feedback) is closer to a more realistic model of how humans iteratively interact with the and then reflect on the environment. The Dyna-Q model [366] describes one such approach to integrating planning and environmental feedback.

Models of physical systems can be seen as estimated state transition models that describe how natural phenomena are likely to occur. Uncertainty Quantification using these physical models can be understood as a means of estimating potential future states of physical systems (planning). These estimated future states can in turn be used to assess the relative utility of future states. For example, consider the construction of a bridge. A simulation of the stability of the bridge may estimate whether or not collapse is likely. This collapse probability can then be used to estimate expected future rewards. If the bridge fails after construction, then the negative reward will be large. If the bridge does not collapse after construction, a desirable future state will be reached and positive rewards for the agent are likely. Risk, in the usual engineering sense of event probability factored by consequences [21, 118], is thus recovered.

Acceptable risk, which seeks to estimate the level of risk tolerated by society, is often used to discuss the possible negative consequences of large engineering projects [95]. Acceptable risk analysis acknowledges that it is not possible to

ensure that all risk can be eliminated in the future by actions in the present. By considering the public perception of risk, designers can (notionally), adjust their utility functions in order to avoid negative repercussions of design failures. In Civil Engineering, failures can often be associated with loss of life. Although effective for risk comparisons between different types of risks, acceptable risk analyses carry certain dangers to the public. In particular, the risk and acceptable risk concepts can be leveraged by unscrupulous actors as a justification for lowered safety standards to save on costs. In this case, the utility functions of the designers, construction firms or investors and those actually at risk by engineering failures may be misaligned. Viewing engineering design from the perspective of sequential decision making processes thus suggests the need for the (actually at risk) public to maintain the means to lower the utility of those in a position to cause them harm.

Sequential decision making processes also provide a justification for the use of effective phenomenological physical models. The resources expended during planning can be related to the Value of Information. Computational or energy costs used to estimate future states and actions potentially reduce the utility of model as these resources cannot be allocated to other tasks. If models of physical systems successfully predict future states with high probability for low computational effort, then they are likely to have a higher utility than a complex model which cannot be used to predict future states easily (despite being a more accurate model). For example, modelling fluid flow as a continuum rather than as individual molecules is an effective model for the purposes of dam spillway design. Given the known physics of water, modelling all water molecules down to the level of subatomic particles within the dam system will be a more accurate model of fluid flows. However, for a large body of water, it will not be possible to estimate the fluid flows through the spillway by simulating all molecules without an enormous computational expenditure. Continuum models of fluid flow are, however, quite effective for the types of flow conditions encountered during dam design. The utility gained by the effective continuum model is far higher in this case as the dam spillway future states can be estimated within a reasonable time frame. The continuum model would be incredibly inaccurate for estimating the actual physical behaviour of the constituent components of water, say, close to the speed of light near a black hole singularity. Such a situation is, however, unlikely to be relevant to an agent designing a dam (that is, the probability for an agent to be near

these states is very small).

Expending effort to estimate actions in unlikely states will have a lower utility than applying effective models which generate reasonably accurate estimates over likely states. From an information theory and Shannon entropy perspective, there is essentially no new information gained by the more complicated model compared to the simple model if, from the perspective of the agent, the physically indistinguishable microstates (say, atomic configurations) do not lead to observably different macroscopic future states [34]. Computational expenditure on physically indistinguishable states is thus also wasteful. If there is a model which yields the same information as another for less effort, then the value of the information gained by the low effort model will be relatively higher. Future state inference using effective models can be seen as a probabilistic estimation problem. In particular, Bayesian Inference can be used to improve numerical methods for simulations as is shown in Chapter 7. As such, Uncertainty Quantification and probabilistic modelling techniques are useful for estimating probable future states for a given model.

2.3 Uncertainty Quantification

When performing planning to facilitate decision making in the real world, simulations of physical systems can be used to aid in predictions of future outcomes. The essential assumption is that the true state of the world is not known (and perhaps not completely knowable depending on ones interpretation of quantum mechanics [31]) but follows some discernible patterns. This thesis will avoid straying too far into the philosophy of randomness (see for example [63, 64]) and instead adopt a pragmatic view that individuals can make inferences about future world states and, by doing so, make judgements in the present that will increase their utility in the future. In the language of Game Theory, simulation can be used as a part of planning in that models that are intended to estimate future states by approximating the true behaviour of the world. These models can be used to facilitate decision making in the present. It is therefore beneficial to consider how Uncertainty Quantification (probabilistic modelling of physical system simulation inputs and outputs) can be used to facilitate planning. In engineering, for example, simulation can be used to predict the suitability of some design and to help estimate the influ-

ence of design changes. Similarly, in scientific work simulation can be used to, for example, suggest the design of future experiments in order to maximise data collection capabilities. Of course, these examples are far from the full range of applicability of simulations in various fields.

Uncertainty Quantification in general refers to the synthesis of probability theory with predictive modelling techniques in order to estimate prediction variability in the presence of uncertainty. Uncertainty Quantification for physical systems uses probability theory to model how the range of outputs from some simulation can be influenced by uncertainty in the inputs to the model. Note that a model may also refer to meta-model analysis, that is, there may be uncertainty as to which set of equations best describes some physical system, for example see [84]. This thesis will be mainly concerned with the problem of estimation of uncertain output quantities given a simulation method and a probabilistic description of input parameter uncertainty. When modelling physical systems, so-called inverse problems (for example estimating material property parameters given known stresses and strains) can also be seen as a problem of estimating unknown outputs given inputs. Figure 2.1 presents a graphical description of physical system Uncertainty Quantification. Chapter 3 discusses suitable models for probabilistic input distributions for physical system Uncertainty Quantification. In particular, random field models of probabilistic spatial correlation structures are detailed.

For clarity, this Section formally defines a number of terms from Uncertainty Quantification. In particular, the *input and output space distributions* refer to the probability distributions that define the uncertainty present in the model. The input space distribution is a given assignment of the relative confidence possessed over a set of parameters. These uncertain parameters are then fed into a model (defined as a part of the problem specification). Given these uncertain parameters and the model, the goal is to estimate the uncertainty in the model outputs (the output space distribution). Finally, one may wish to estimate the expected value of a *Quantity of Interest* (QoI). A QoI, defined below, is typically taken to be the expected value of some function of the output space to the model. As such, calculating QoI's requires an estimate of the output space probability distribution. Further detailed discussion on the structure of Uncertainty Quantification problems is presented in Chapter 6.

This thesis will focus on Partial Differential Equation (PDE) models as these are sufficient to describe continuum mechanics [250]. For civil engineering applications, continuum mechanics is crucial as it can be used to define both fluid flow and solid deformations. Common solution techniques for PDE problems include Finite Element, Finite Difference and Finite Volume methods [198]. Beyond continuum mechanics, PDE's can also be used to model a wide range of physical behaviours, transport phenomena [386], electrodynamics and gravitational forces [126]. Other common simulation methods, such as N-body simulation [126] and lattice-Boltzmann particle type methods [69], are not considered directly in this thesis. This thesis will primarily work with Finite Element PDE solutions. However, based on the arguments presented in Chapter 6, the Uncertainty Quantification techniques presented are sufficiently general that they should be able to be applied to any typical simulation methodology. This is further supported by the probabilistic interpretation of solving equations presented in [175] and expanded on in Chapter 7.

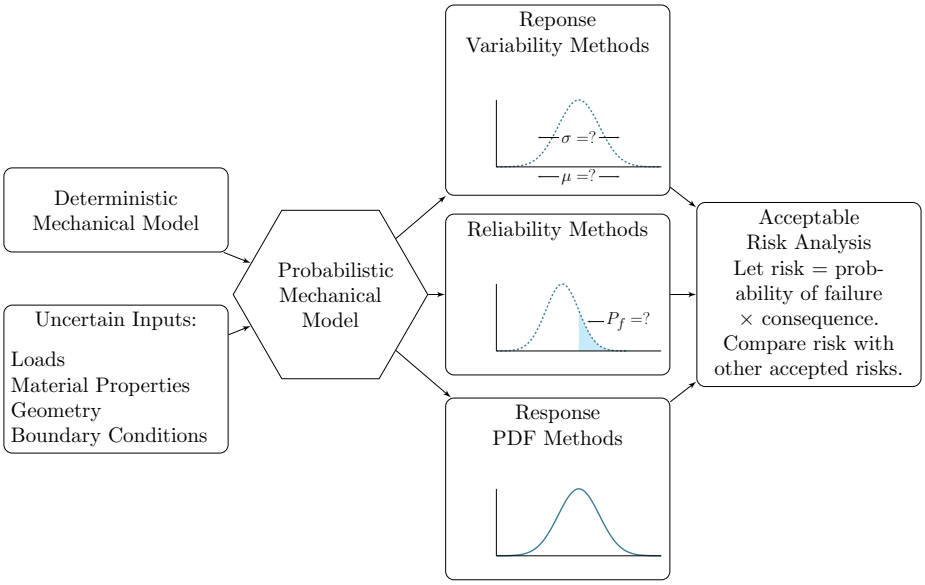


Figure 2.1: Graphical description of Uncertainty Quantification for typical engineering problems. Acceptable risk analysis considers project risks relative to societally acceptable risks. Adapted from [356].

2.3.1 Input and output models and distributions

The input and output spaces to a problem represent the domains on from which full problem specifications and solutions can be drawn. For example, the input space for an uncertain heat equation might be the selection of thermal conductivities at every point within the spatial problem domain. The output space would then, in this example, represent a possible solution defined at every point within the problem spatial domain.

Definition 2.3.1. *Input and output space*: Let the sets X and Y be the *input space* and *output space* respectively. Let points within the input and output spaces be represented by $x \in X$ and $y \in Y$ respectively. $\triangle$

In order to carry out Uncertainty Quantification, it is necessary to consider probability measures over the input and output spaces.

Definition 2.3.2. *Input, output and joint input-output probabilities*: Given X and Y, let $P(X)$ and $P(Y)$ be the *input probability density* and *output probabilities* respectively. Then let $P(X, Y)$ be the *joint input-output probability density.* $\triangle$

Deterministic mechanical models can be specified in terms of a residual functional. The residual functional, discussed extensively in Chapter 6, defines a solution of a set of equations as the output, y, for which the error functional is a minimum. The error functional is defined as:

Definition 2.3.3. *Residual functional*: Given a function space $\mathcal{H}$ let:

$$H : X \times Y \longrightarrow \mathbb{R}^+ \tag{2.5}$$

such that

$$H(y|x) \mapsto \epsilon \in \mathbb{R}^+ \tag{2.6}$$

be the *residual functional.* Other names for this functional include the *error*, *loss* and *energy* functional. The *Hamiltonian* is also used to refer to equation (2.5) in the discussion in Chapter 6. $\triangle$

2.3.2 Quantities of Interest

So far this Section has specified the general problem framework. The final element needed for Uncertainty Quantification is the definition of Quantities

of Interest [342], the expectation value of functions of the output space that are to be estimated. These functions are referred to as Quantity of Interest functions.

Definition 2.3.4. *Quantity of Interest (QoI) function*: A *Quantity of Interest function*, ψ, is a function of the output space:

$$\psi : Y \longrightarrow \Psi \tag{2.7}$$

where Ψ is the image space of ψ. △

Examples of such quantities might include the maximum displacement in a stress-strain problem, the average pressure in an instance of the Navier-Stokes equations, the gradient of the temperature in the heat equation or whatever other quantity that one may wish to estimate. As a particular example, given a solution to the steady state heat equation $y = u(s)$ over the spatial domain S, consider the QoI function that calculates the maximum and minimum temperature, then $\Psi \equiv \mathbb{R}^2$ such that

$$\psi(u(s)) = \begin{bmatrix} \max\limits_{s \in S}(u(s)) \\ \min\limits_{s \in S}(u(s)) \end{bmatrix}$$

One goal of Uncertainty Quantification is to calculate Quantities of Interest given QoI functions and the output space probability density [342]:

Definition 2.3.5. *Quantity of Interest (QoI)*: A *Quantity of Interest* under $P(y)$, $\langle \psi(y) \rangle_{P(Y)}$, is the expectation value of a quantity of interest function, ψ given $P(y)$:

$$\langle \psi(y) \rangle_{P(Y)} := \mathbb{E}\left[\psi(y)\right]_{P(Y)} = \sum_{y \in Y} \psi(y) P(y) \tag{2.8}$$

△

It is noted here that the Quantity of Interest function can be defined over the joint input-output space, i.e. $\psi : X \times Y \to \Psi$. This extension can be made without the loss of generality for the results in this thesis. However, as the focus of this thesis is on particular problems in Uncertainty Quantification, it is more useful from an expository perspective to include only ψ acting on Y within as this is the most typical case.

2.3.3 Sources of uncertainty in physical systems

A probabilistic formulation of sequential decision making naturally induces a probabilistic model of planning. The relevance of Uncertainty Quantification for physical systems can, however, be related more directly to physical intuition. It is debatable as to whether or not the physical universe is fundamentally probabilistic or deterministic [294]. Regardless of any fundamental randomness in the universe, there are thermodynamic limitations imposed on the ability to collect, store and process information about the physical world [245, 124]. Information processing requires energy expenditure (which is why computers need to be powered) and information storage requires space. Assuming that there is a finite amount of both energy and particles in the universe, it follows that computation is limited physically. Estimates of fundamental limits to computation are discussed in [245]. It is worth noting that modern computers are very much weaker than the current estimates of the upper bounds of computational power imposed by physical constraints. Uncertainty Quantification, then, emerges as a tool for dealing with computational constraints when modelling physical systems.

These constraints can be viewed in terms of local resource conservation for utility maximisation using a probabilistic planning model of physical state estimation, as discussed in Section 2.2. This viewpoint does not, however, require that computational resources are fundamentally limited, only limited from the perspective of an agent. By introducing the thermodynamic limits to computation, Uncertainty Quantification becomes required at a fundamental level. If computational resources are limited, and future physical state estimation requires computational resources, then it becomes necessary to use Uncertainty Quantification to make useful estimates about future states. As will be discussed in Chapter 6, even deterministic models can be viewed as probabilistic models by framing these problems as Bayesian Inference. In particular, deterministic models that estimate the behaviour of a physical system to within a numerical tolerance limit can be viewed as Maximum Likelihood Estimates taken from probabilistic models of expected future states.

Less abstractly, the sources of uncertainty when modelling physical systems are primarily the model selection error and the uncertainties in the model inputs. Examples of model selection problems include deciding upon a material constitutive model for stress-strain analysis or allowing for discontinuous functions

to represent a solution when solving, say, a fracture problem. Uncertain inputs to a given model are a further cause of uncertainty. For physical systems, as shown in Figure 2.1, typical uncertainties include unknown material property parameters, unknown forcing and unknown boundary conditions. The ability to collect data is frequently limited when modelling real world phenomena such as the shear strength of soils. Probabilistic input models can be seen to be prior distributions on the possible range of values of uncertain inputs. The prior distributions represent subjective beliefs about the likely input parameter values. New knowledge, such as field testing, can be included into the belief model (most rigorously by Bayesian updating). By restricting input uncertainty, the estimated output probability distribution from a physical system will be altered. Input uncertainty models can thus be used to restrict the range of possible outputs from a model to only those that are likely given the likely model inputs. This is the implicit procedure adopted for the majority of practical engineering design. Uncertainty Quantification allows for such ad-hoc procedures to be understood rigorously.

2.3.4 Approaches to the solution of Uncertainty Quantification problems

Solving problems in Uncertainty Quantification is typically computationally demanding as simulations of physical phenomena (such as PDE solvers) must be augmented by probabilistic descriptions of the inputs and outputs to such a simulator. This augmentation typically requires that a large number of additional simulations must be evaluated in some form. This Section discusses the structure of Uncertainty Quantification problems and reviews existing approaches to Uncertainty Quantification. The traditional approach (termed here the *bottom up approach*) to Uncertainty Quantification research in physical systems is compared and contrasted with what will be termed the *top down approach*. The bottom up approach refers to the traditional development strategy adopted in numerical methods research, that is, iterative improvements and refinements on existing techniques. The top down approach instead considers, abstractly, the ideal method for Uncertainty Quantification and then attempts to work 'downwards' towards practical implementations. The top down approach will be used to provide guidance to the numerical methods explored in the later Chapters of this thesis and to suggest directions

for future research. The brief outline of existing Uncertainty Quantification methods presented in this Section will be expanded on in the later Chapters of this thesis.

2.3.4.1 Top down perspectives on Uncertainty Quantification

Uncertainty Quantification can be viewed as a part of planning for a sequential decision making process and, as such, a solution to the more general decision process problem would constitute a solution to physical system Uncertainty Quantification. However, because Uncertainty Quantification for physical systems concerns real-world decision making processes, this would require solving the full Artificial General Intelligence (AGI) problem, described in [323]. Assuming that solving AGI is too difficult for this thesis, it is useful to restrict the top down approach to consider the problem of estimating unknown output density and QoI values given a physical model and input space distribution. Estimating appropriate models of physical behaviour given observations from nature is essentially the scientific method which can itself be framed as Bayesian Inference (or induction) [354] and will not be considered in detail. Estimating parameter values and their distribution, required to generate the necessary input distributions, is also a typical application of statistics to science and engineering and will not be extensively discussed. The QoI and output space density estimation problems will be the key concerns of this thesis.

Output space density estimation requires an integral over the inputs to the physical model to be evaluated. QoI estimates require expectations over estimated output space densities to be calculated. In the Bayesian setting, this is referred to as prediction. As such, both tasks are problems of integration and so all of Uncertainty Quantification as defined in this Chapter (given a physical model) can then be seen as an integration problem. Integration, itself, can be considered from a Bayesian inference perspective. Assuming that an integral converges, the task is to infer the unknown value of the integral. With reference to the discussions on utility maximisation and the Value of Information in this Chapter, the optimal integration method would be able to estimate the unknown value of the integral in question with a high probability for as little computational effort as possible. The task, then, of research into numerical Uncertainty Quantification methods is to develop efficient methods for inte-

gration. The challenge becomes developing methods that are effective for the very complicated integrals encountered when high dimensional, spatially correlated joint distributions (for example, random fields which are discussed in Chapter 3) are combined with, say, PDE models of physical behaviour.

2.3.4.2 Integration approaches for Uncertainty Quantification

Returning to the bottom up approach to Uncertainty Quantification, this Section briefly introduces the two major integration techniques for physical system Uncertainty Quantification used in practice, Monte Carlo Simulation and Series Expansion Methods. Monte Carlo Methods evaluate integrals by repeatedly evaluating the function to be integrated at randomly sampled locations. Series Expansion methods, on the other hand, utilise functional analysis techniques to expand the functions of interest in terms of simpler functions. Given such a series expansion, complicated integrals can be evaluated by the summation of a number of simpler integrals. More detailed discussions of these methods are also presented in later Chapters of this thesis.

The dimensionality of the functions to be integrated represents the primary difficulty when attempting to solve Uncertainty Quantification problems. Spatial randomness models induce infinite (or very high) dimensional probability distributions. Then, Uncertainty Quantification for models with spatial randomness requires evaluating these high dimensional integrals. For example, the uncertain inputs to a stress-strain simulation may involve uncertainty in the material elastic stiffness. If the elastic stiffness is modelled so that each point within the material domain is drawn from some probability distribution, the dimension of the induced probability measure of the spatial domain will have dimension related to the total number of points in the spatial domain.

The challenges of high dimensional integration are a manifestation of the *curse of dimensionality* [32]. The volume of a mathematical space increases exponentially with the number of dimensions present. Brute force numerical integration would require that the function to be evaluated is computed at all points within the integration domain. This is obviously intractable for all but the most simple problems. Quadrature methods integrate functions by approximating a function of interest with simpler functions with a priori known integral values [111]. This allows for generalisation (interpolation) be-

tween known function values. Unfortunately, standard numerical quadrature integration methods, such as Gaussian quadrature, fail in high dimensional spaces. The number of integration points required to evaluate high dimensional integrals by standard quadrature methods increases exponentially with the dimensionality of the problem and as such renders this form of integration computationally intractable [179]. The sparse grid method [130] is a high dimensional quadrature technique that addresses the curse of dimensionality by a careful choice of quadrature points. In particular, the tensor product of one dimensional quadrature rules are combined carefully in such a way that the truncation error increases slowly with increasing dimensionality. Applications of sparse grid methods to Uncertainty Quantification are presented in [105, 231]. As these methods can be difficult to implement and alternative method (discussed below) are available, they are not considered further in this thesis. The various analyses presented in Chapters 4 and 5 could, however, potentially be modified to use sparse grid techniques. As will be expanded on in Chapter 5, one of the goals of this thesis is to avoid hand design of the structure of the numerical algorithms used in favour of self-improving algorithms. Examples of self-improving algorithms for problems related to Uncertainty Quantification are detailed in Chapters 5 and 7.

One approach to solving high dimensional integrals is to reduce the effective dimensionality of the problem. Series Expansion methods, such as the Spectral Stochastic Finite Element Method (SSFEM) [141, 356], perform dimensionality reduction by projecting high dimensional functions onto a series of lower dimensional basis functions. Techniques related to Series Expansion methods are Polynomial Chaos methods, generalised Polynomial Chaos and Orthogonal Series Expansions [351]. A key tool in these series expansion methods is the Karhunen-Loève Expansion of a random field (discussed in detail in Chapter 3). These methods project the random, spatially correlated, response function of the solutions of a physical simulation with random inputs onto a basis expansion of a high dimensional probability distribution. SSFEM is one example of such a method. The lower dimensional problem over the basis functions can be solved more efficiently than the original high dimensional problem. These methods can be very effective, in particular when considering the first and second moments of problems with Gaussian uncertainty.

As will be discussed in Chapter 4, these SSFEM-type methods can be difficult

to apply for nonlinear PDEs. Additionally, these methods can be unsuitable for rare event simulation. By contrast, Chapter 4 demonstrates that Monte Carlo Simulation (MCS) can be used effectively in these instances. Monte Carlo Simulation evaluates high dimensional integrals of the form:

$$I(X) = \int_{\Omega} f(x) d\mu(x) \tag{2.9}$$

by the approximation:

$$I(X) \approx \hat{I}(X) = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{2.10}$$

where each x_i is a sample from the probability distribution defined by $d\mu(x)$. Moving away from a probabilistic perspective, a special case of Monte Carlo Integration is integrating functions over spaces without reference to a probability density over the space. In this case, the integration points are sampled uniformly at random from the function domain. By the law of large numbers and the Central Limit Theorem (see Chapter 4), the value of the integral is given by the distribution:

$$\mathcal{N}\left(\hat{I}(X), \frac{\sigma^2}{N}\right) \tag{2.11}$$

after N simulation steps. The curse of dimensionality is avoided by Monte Carlo Integration as error on the estimated integral solution, $\frac{\sigma^2}{N}$, is independent of the integral domain dimension. Essentially, Monte Carlo can be viewed as a randomised dimensionality reduction technique as more likely function values are more likely to be sampled from more often. Monte Carlo Simulation is useful for Uncertainty Quantification in that, given a method to sample from an input distribution and a deterministic numerical solver, the output space distribution can be approximated by repeated sampling. Markov Chain Monte Carlo (MCMC) can be a more efficient sampling methodology than direct MCS [55]. Applications of Monte Carlo Simulation and MCMC for the probabilistic analysis of a nonlinear stress-strain problem are presented is presented in Chapter 4 based on work published by the author [157]. Quasi-Monte Carlo (QMC) methods are an alternative method for improving MCS by sampling so-called low-discrepancy points [57, 227]. QMC methods attempt to improve on the convergence rate of regular MCS by sampling from carefully spaced points over the function domain, capturing more of the function

variability with a higher probability. Latin Hypercube methods are related to QMC in that they attempt to sample from the function domain in a structured way [197]. Low discrepancy sampling can cause degraded performance in some circumstances (see [227]) and are not considered in this thesis. Incorporating low discrepancy sampling with the methods explored in subsequent Chapters would be a useful avenue for future research.

The basic MCS procedure for solving an Uncertainty Quantification method is to repeatedly sample from the input distribution, pass these inputs to a numerical PDE solver and then evaluate the PDE solutions. This process is repeated until satisfactory integral convergence is obtained. Although this method is simple, Monte Carlo methods face a serious deficiency in that the function to be integrated must be evaluated a large number of times. When the function in question is a PDE solution, actually evaluating these solutions is computationally intensive. Moreover, high probability points will be sampled from frequently, meaning that the solution of only very slightly different PDE problems will be calculated repeatedly. This is computationally wasteful and it would be more efficient to store the results of known PDE evaluations in such a way that the solution for new, unseen inputs can be estimated based on prior knowledge. To reduce the computational burden of MCS, Chapter 5 introduces an Artificial Neural Network based surrogate model. The surrogate model learns the mapping from inputs to outputs and how to generalise solutions to unseen input instances. This surrogate model can then be integrated by MCS far more cheaply than the PDE solver. This reduces the time taken for MCS integration for computationally expensive sampling. Chapters 6 and 7 present theoretical and practical developments towards a full probabilistic interpretation of both the uncertain problem inputs and the uncertain problem solution. The goal of combining these techniques is to enable the development of self-improving algorithms in the future.

2.4 Conclusions

This Chapter formally introduced Uncertainty Quantification from a Bayesian perspective, as well as the probabilistic concepts required to appreciate this perspective. In particular, the interpretation of Uncertainty Quantification as a part of a sequential decision making process was described. This framing is

useful in that it allows for various aspects of Uncertainty Quantification to be understood more clearly. Engineering and scientific design can be understood as part of a decision making process that seeks maximise some utility function. The simulation of physical systems is a form of planning. The utility of a simulation procedure can be understood, with reference to Value of Information theory, in terms of the information gained versus the computational cost of the simulations. By introducing Uncertainty Quantification and probabilistic physical simulation, variability in the physical system to be simulated can be accounted for by computing the probabilities for various outputs from a system. Using Uncertainty Quantification, engineering and scientific design can be fit into a more universal decision making framework that rigorously assesses risks associated with design and construction processes.

Chapter 2 Appendix: Probability Theory and Bayes Theorem

As this thesis is concerned with Uncertainty Quantification, the use of probability theory is an essential component. As such, a number of standard formal definitions from probability and measure theory are given for notational clarity and consistency. Definitions relating to Bayes Theorem and Bayesian updating are also formally stated. The reader is assumed to have some familiarity with probability theory more generally. Note that in this thesis, all spaces considered are either discrete or continuous as appropriate. Summation symbols denote either an integral on a continuous space (in the sense of Lebesgue measure) or a sum on a discrete space (in the sense of counting measure) as necessary. See [168, 309] for more details on measure theory and Lebesgue integration. The reader, however, should understand that although at times summation symbols (rather than integrals) are used for convenience the results presented are not restricted to discrete probability mass functions on discrete spaces. Further, note that all function spaces are assumed to satisfy the conditions specified in [355]. The notation $\|a - b\|$ is intended to signify a norm of the difference of two items a and b appropriate for the chosen working space.

2.5.1 Probability and Measure Theory

Probability theory is, typically, formally based on measure theory. Measure theory describes how so-called *measures* (real valued functions) can be assigned to elements of sets (in a set theoretic sense). As well as helping to formalise probability theory, measure theory is crucial for formally defining integration theory. A number of definitions relevant to the developments in this thesis are given in this Section.

Definition 2.5.1. *Measure space:* From §17 in [168], a *measure space* is a three element tuple (Ω, Σ, μ) where:

- The tuple (Ω, Σ) is termed a *measurable space* such that:
- Ω is an arbitrary non-empty set.
- Σ is a σ-algebra, where $\Sigma \subseteq \mathcal{P}(\Omega)$ such that:
  - $\Omega \in \Sigma$
  - Σ is closed under complements: if $A \in \Sigma$, $\Omega \setminus A \in \Sigma$
  - Σ is closed under countable unions: if $A_i \in \Sigma$ for $i = 1, 2, \ldots$ then also $\cup_{i=1}^{\infty} A_i \in \Sigma$
- $\mu : \Sigma \to [0, 1] \in \mathbb{R}^+$ is a *measure*, a function from Σ to $\mathbb{R}^+$.

$\triangle$

Note that a *measurable space* is, following Definition 2.5.2, a measure space without a measure assigned, that is the pair (Ω, Σ).

Definition 2.5.2. *Probability space:* From §17 in [168], a *probability space* is a three element tuple (Ω, Σ, P) where:

- (Ω, Σ) is a measurable space where:
  - Ω is termed the *sample space* of all possible *outcomes*.
  - Σ is a σ-algebra, consisting of *events* which are sets consisting of zero or more outcomes.
- $P : \Sigma \to [0, 1] \in \mathbb{R}^+$ is the *probability measure*, which is a measure such that:

– P is countably additive: if $\{A_i\}_{i=1}^{\infty} \subseteq \Sigma$ is a countable collection of pairwise disjoint sets, then $P(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$.

– $P(\Omega) = 1$. Note that this is sometimes referred to as the *law of total probability* [403].

△

Definition 2.5.3. *Measurable map:* From [168], given two measurable spaces (A, Σ_A) and (B, Σ_B) a function $f : A \to B$ is a *measurable map* if for every $C \in \Sigma_B$, the inverse image satisfies $f^{-1}(C) \in \Sigma_A$. △

Throughout this thesis, the term *probability distribution* is also used to refer to a probability measure on a space. Where not stated, the σ-algebra is assumed to be the maximal σ-algebra on the underlying space [168].

The Dirac delta function will be required. It is defined here based on [309] and §2.1 of [35]:

Definition 2.5.4. *Dirac delta function*: In this thesis the *Dirac delta function* is taken to mean the *Dirac measure* which is a measure δ on a set X (with any σ-algebra of subsets of X) for $x \in X$ and a measurable set $A \subseteq X$ by

$$\delta_X(A) := \chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \neq A \end{cases} \tag{2.12}$$

In the case that X is a single element, $x' \in X$, the Dirac measure may be written:

$$\delta(x - x') := \delta_{x'}(A) \tag{2.13}$$

△

For example, on a real valued continuous space the Dirac delta function can be used to select a particular value of a function:

$$\int_X f(x)\delta(x)dx = f(0)$$

The Dirac measure can also be interpreted as a Kronecker delta for a discrete

sum:

$$\delta_{ij} := \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

so that, for example:

$$\sum_i \delta_{ij} f(x_i) = f(x_j)$$

To actually make use of probability theory in this thesis, a number of additional definitions are required which are given below. In particular, a definition of conditional probability is required. Although there are several possible definitions of conditional probability this thesis will, for simplicity, adopt the standard Kolmogorov definition of conditional probability and avoid some of the difficulties faced when defining conditional probabilities on continuous spaces. Products and sums of probabilities are also discussed. For a more full measure theoretical treatment of product measures and conditional probability see [168, 50].

Definition 2.5.5. *Conditional probability ([216]):* Let (Ω, Σ, P) be a probability space. Let A and B be events Σ with $P(B) > 0$. The conditional probability of A given B is defined as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{2.14}$$

△

Reorganising the definitions of conditional probability yields $P(A|B)P(B) = P(A \cap B)$. As per Theorem 1.3 in [224], from the reorganised conditional probability rule and the definition of probabilities on intersections of events the *chain rule for probabilities* or *product rule* is defined as:

Definition 2.5.6. *Chain rule for probabilities and independence:* Let (Ω, Σ, P) be a probability space. Let $A = \{A_1, A_2, \cdots A_n\}$ be a set of events in Σ such that $P(A) > 0$. Then $P(A_1, A_2, \cdots, A_n) := P(A_1 \cap A_2 \cap \cdots \cap A_n)$ and:

$$\begin{aligned} P(A_1, A_2, \cdots, A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2, A_1)\times \\ \cdots \times P(A_n|A_{n-1}, \cdots, A_2, A_1) \end{aligned} \tag{2.15}$$

the events $\{A_1, A_2, \cdots A_n\}$ are *independent* if

$$P(A_1 \cap A_2 \cap \cdots \cap A_n) = P(A_1)P(A_2)P(A_3)\cdots P(A_n) \tag{2.16}$$

△

Next, following §1.5.2 of [224], define a partition of a probability space as:

Definition 2.5.7. *Partition of a probability space:* Let (Ω, Σ, P) be a probability space. Let $B = \{B_1, B_2, \cdots B_n\}$ be a *partition* of Ω where a partition is set of events in Σ that are pairwise disjoint (non-overlapping) that cover the space Ω (that is, $\bigcup_{i=1}^{n} B_i = \Omega$). It follows that $P(B) = 1$. △

Given a partition, following Theorem 1.4 in [224] define the *law of total probability for an event* as:

Definition 2.5.8. *Law of total probability for an event:* Let (Ω, Σ, P) be a probability space. Let $B = \{B_1, B_2, \cdots B_n\}$ be a partition of Ω. Then:

$$P(A) = \sum_{i=1}^{n} P(A|B_i)P(B_i) \tag{2.17}$$

△

The measure theoretic definition of a random variable is a map from a probability space to a measure space which, following §1.3 of [317], is formally defined as:

Definition 2.5.9. *Random Variable:* Let (Ω, Σ, P) be a probability space. Let $(A, \mathcal{A})$ be a measurable space. Then an $(A, \mathcal{A})$-valued random variable (or A-valued random variable) is a function X defined by:

$$X : \Omega \to A \tag{2.18}$$

such that for all $a \in \mathcal{A}$, $X^{-1}(a) \in \Sigma$ (the subset a has preimage under X in Σ) with $X^{-1}(A) = \{\omega : X(w) \in a\}$. △

By defining a random variable in this way, the probability of any event in $a \in \mathcal{A}$ is defined by the probability measure calculated over the region defined by $X^{-1}(a)$ on the original probability space. This is a typical pushforward operation to transfer some function from one space to another (see [125]). For real-valued random variables (that is, $X : \Omega \to \mathbb{R}$), this definition can be extended to the usual definition of a real-valued random variable in terms of a cumulative distribution function. A real-valued random variable $X(\omega)$ on (Ω, Σ, P) can be shown to be given by:

$$\{\omega : X(\omega) \leq v\} \in \Sigma \quad \forall v \in \mathbb{R} \tag{2.19}$$

because $\{(-\infty, v] : v \in \mathbb{R}\}$ generates a (Borel) σ-algebra on $\mathbb{R}$ and $\{\omega : X(\omega) \leq v\} = X^{-1}((-\infty, v])$ (see §2 of [302]). What this means is that a function, the cumulative distribution function $F_X(x)$, can be defined from equation (2.19). First, note that $P(\{\omega : X(\omega) \leq x\}) = X^{-1}((-\infty, x])$ which can be written more concisely as:

$$P(X \leq x) := P(\{\omega : X(\omega) \leq x\}) = X^{-1}((-\infty, x]) \tag{2.20}$$

from which the cumulative distribution function can be defined:

$$F_X(x) = P(X \leq x) \tag{2.21}$$

The the *joint probability* of two (or potentially more) random variables is defined as follows:

Definition 2.5.10. *Joint Probability:* Let (Ω, Σ, P) be a probability space. Let X and Y be random variables. Then the joint probability of X and Y is defined as:

$$P(X = x \text{ and } Y = y) = P(Y = y|X = x)P(X = x) \tag{2.22}$$
$$= P(X = x|Y = y)P(Y = y) \tag{2.23}$$

△

which can be extended to multiple random variables, $X_1, X_2, \cdots X_n$ by application of the probability chain rule in Definition 2.5.6:

$$\begin{aligned} P(X_1 = x_1, \cdots, X_n = x_n) = & \\ & P(X_n = x_n|X_{n-1} = x_{n-1}, \cdots, X_1 = x_1)\times \\ & \quad \cdots \times P(X_2 = x_2|X_1 = x_1)P(X_1 = x_1) \end{aligned} \tag{2.24}$$

With the appropriate measure theoretic definitions in place, it is possible to define integration. A complete exposition of measure theoretic integration would be more detailed than is necessary for this thesis and the interested reader is referred to [168], [317] (in particular §4), §3 of [285] and §8 of [217]. Consider the measure space (Ω, Σ, μ) and a partition of Ω. A partition, from Definition 12 in §1.3 of [317] is a collection of sets $\{A_i\}_{i=1}^n$ such that $A_i \cup A_j = \emptyset$ is $i \neq j$ and $\sum_{i=1}^n A_i = A$. Next, define a simple random variable (from

Definition 13 in §1.3 of [317]) as $X : \Omega \to \mathbb{R}$ by $X = \sum_{i=1}^{n} \alpha_j \chi_{A_j}$ over the (countably infinite measurable) partition $\{A_i\}_{i=1}^{n}$ with that $\alpha_j \in \mathbb{R}$ where χ_{A_j} is the indicator function (equivalent here to the Dirac measure) over A_j. Then, from §23 of [168], the simple function X is said to to *integrable* if $\mu(A_j) < \infty$ for all A_j for which $\alpha_j \neq 0$. Symbolically, the *integral* of X is denoted:

$$\int_{\Omega} X(\omega) d\mu(x) \tag{2.25}$$

or, equivalently by $\int X d\mu$. The integral is defined by:

$$\int X d\mu = \sum_{j=1}^{n} \alpha_j \mu(A_j) \tag{2.26}$$

Further, for some subset $E \in \Omega$ the integral over E is defined by:

$$\int_{E} X d\mu = \int_{\Omega} \chi_E X d\mu \tag{2.27}$$

From §8 Definition 29.1 of [217], the *Lebesgue integral* defines integration by partitioning the range of a function and is defined as follows:

Definition 2.5.11. *Lebesgue integral:* Let f be a simple Y-valued measurable function on (Ω, Σ, μ). Let y be a partition of Y so that f has no more than a countable number of distinct values, $y_1, y_2, \cdots, y_n, \cdots$. Then the *Lebesgue integral* over E is denoted as:

$$\int_{E} f(\omega) d\mu(\omega) := \sum_{i} \mu(A_i) \tag{2.28}$$

where

$$A_i := \{\omega : \omega \in A, f(\omega) = y_i\} \tag{2.29}$$

provided that the series in equation (2.28) exists and is absolutely convergent. △

See [217] for a definition of "absolutely convergent". For the standard case of integration of real valued functions over the number line encountered in typical engineering mathematics, the Lebesgue integral is equivalent to the Riemann integral (see §8.30.3 of [217]).

This definition of an integral is sufficient to define the *expectation value* or

(expected value) operator of a random variable defined, from Theorem 13 in §4.3 of [317], as:

Definition 2.5.12. *Expectation value:* Given a probability space (Ω, Σ, P), let X be an A-valued random variable on (Ω, Σ, μ). Then the *expectation value* of X is defined as the Lebesgue integral of X:

$$\mathbb{E}[X] := \int_{\Omega} X dP = \int_{\Omega} X(\omega) dP(\omega) \tag{2.30}$$

if this integral exists. Other notation for the expectation value includes $\langle X \rangle$ or $\mathbb{E}_P[X]$ and $\langle X \rangle_P$ where the subscript denotes the probability measure. $\triangle$

2.5.2 Bayesian Inference and Bayesian Probability

2.5.2.1 Introduction

Bayes Theorem is an essential component for understanding how Uncertainty Quantification can be incorporated into a rigorous decision making framework. Note that the grammatically correct Bayes' Theorem (with the apostrophe) will not be used in this thesis for notational convenience. Starting with a definition of Bayes Theorem, this Section also defines concepts from *Bayesian Inference*. Bayesian Inference uses Bayes Theorem to formulate how observed data can be used to reason about the value of *Latent variables* (or Hidden variables). This is done by using a set of initial, subjective beliefs regarding the probability of various outcomes (the prior distribution over hypotheses) which can be updated (to the posterior distribution) given additional observations (data) by using the joint distribution over both data and hypotheses as a representation of known information. This type of model can be represented by a *Bayesian Network* [323, 215], a directed graph which indicates conditional dependencies between the joint distribution random variables. The relationship between Bayesian Inference and *Maximum Likelihood Estimation* is briefly discussed, with a more detailed discussion presented in Chapter 6. Bayesian Inference can be extended to include time as a variable by considering how the probability of hidden states changes over time. This leads to notions of *filtering*, *smoothing*, *prediction* and *learning*. These concepts are demonstrated in this Section by a *Hidden Markov Model*. This framework provides a means to understand probabilistic numerical methods [175]. Such methods are

utilised in Chapter 7 to derive an Expectation-Maximisation based algorithm for Element Free Galerkin adaptive basis function refinement.

At the core of these ideas lies Bayes Theorem which relies on the simple observation that the joint probability of two different events is related to the conditional distributions of one event given another. Bayes Theorem is formally defined, from Theorem 1.5 [224], as:

Definition 2.5.13. *Bayes Theorem:* Let (Ω, Σ, P) be a probability space. Let $A \in \Sigma$ be an event with $P(A) > 0$. Let $B = \{B_1, B_2, \cdots B_n\}$ be a *partition* of Ω where a partition is set of events in Σ that are pairwise disjoint (non-overlapping) that cover the space Ω such that $P(B) = 1$. Bayes Theorem is defined as:

$$P(B_j|A) = \frac{P(B_j|A)}{\sum_{i=1}^{n} P(A|B_i)P(B_i)} \tag{2.31}$$

△

where Bayes Theorem is derived by combining the law of total probability for an event (Definition 2.5.8) with the definition of conditional probability:

$$P(B_j|A) = \frac{P(A \cap B_j)}{P(A)} \tag{2.32}$$

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{P(A)} \tag{2.33}$$

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_{i=1}^{n} P(A|B_i)P(B_i)} \tag{2.34}$$

Bayesian Inference, in the most simple form, uses observed data (the evidence), d, to estimate hypothesis, h, from a space of possible hypotheses, H, best models the data:

$$P(h|d) = \frac{P(d|h)P(h)}{\sum_{h \in H} P(d|h)P(h)} \tag{2.35}$$

where:

- d is the current *evidence*.
- h is a *hypothesis* in the *hypothesis space* H.
- $P(h)$ is the *prior distribution*.
- $P(h|d)$ is the *posterior distribution*.
- $P(d|h)$ is the *likelihood*.

- $P(d) = \sum_{h \in H} P(d|h)P(h)$ is the *marginal likelihood* or *model evidence.*

Bayesian Inference aims at calculating the probability that some hypothesis h is correct. Hypotheses may also be referred to as models. The prior distribution, $P(h)$ defines a subjective belief that the hypothesis h is correct prior to observing the latest data, d. The posterior distribution, $P(h|d)$, defines the belief in the hypothesis h after incorporating knowledge of the latest observation. The likelihood term, $P(d|h)$ is used as a part of the calculation and defines how probable the observed data is for a given model. Intuitively, Bayes Inference says that observations with a low likelihood given h will cause the posterior probability for that hypothesis to decrease. Conversely, observations with a high likelihood under h will increase the probability of h under the posterior distribution. The marginal likelihood term is also referred to as the normalisation constant and reweights the posterior probabilities under all hypotheses to sum to one. For many inference problems, this is not necessary as only the relative probability of the various hypotheses is needed. In particular, *Maximum a Posteriori* (MAP) Estimation selects the hypothesis with the greatest posterior probability as the correct hypothesis and disregards all others [275]. Maximum Likelihood Estimation (MLE) is a special case of MAP that uses a uniform prior over the hypothesis space [46, 275].

The joint distribution over variables can be represented using graphs. There are a variety of what are termed probabilistic graphical models and detailed treatments are provided in [215, 292]. For example, the joint distribution can be represented as an undirected graph where unlinked nodes represent conditionally independent variables. A Bayesian Network is a graphical model that uses acyclic directed links between nodes to represent conditional dependencies between random variables. Links in the network point from *child* nodes to *parent* nodes where conditional dependencies are present in factorisations using the chain rule for probability (Definition 2.5.6) of a joint distribution, as in $P(parent|child)$.

Consider the following example. Let $P(a_4, a_3, a_2, a_1)$ be a joint distribution over each variable a_i. Using the chain rule for probability factorise this distribution as $P(a_4|a_3, a_2, a_1)P(a_3|a_2, a_1)P(a_2|a_1)P(a_1)$. Further, it is given that, due to conditional independence of a_4 from a_1 and a_2, the factorisation can be written as $P(a_4|a_3)P(a_3|a_2, a_1)P(a_2|a_1)P(a_1)$. Let each variable a_i be a node in the graph. Then let $e_i : \{a_1, \cdots, a_n\}$ denote a set of directed links starting

at a_i and terminating on nodes $a_1, \cdots, a_n$. Then the factorisation:

$$P(a_4|a_3)P(a_3|a_2, a_1)P(a_2|a_1)P(a_1) \tag{2.36}$$

for the example can be expressed as the set of links:

- $e_1 : \{a_2, a_3\}$
- $e_2 : \{a_3\}$
- $e_3 : \{a_4\}$
- $e_4 : \varnothing$

where $\varnothing$ denotes the empty set. This example Bayesian Network is shown in Figure 2.2. Graphical models can be useful for visualising the structure of dependencies in probabilistic algorithms. Other types of graphical models, such as factor graphs, are discussed in [215].

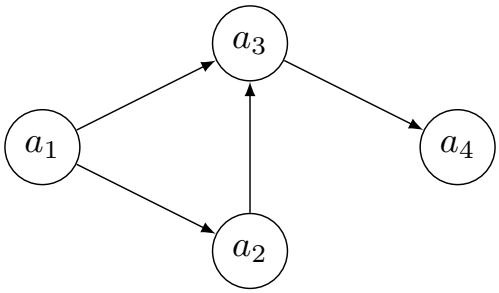


Figure 2.2: Example of Bayesian Network showing the distribution in equation (2.36).

2.5.2.2 Bayesian Inference over time

Static Bayesian Inference aims to estimate the probability over hidden states given a fixed set of observations. Time dependencies can be introduced by indexing states and observations with a time variable (either continuous or discrete). The *belief state* represents the current belief about the hidden states. In such a model, states change in time by a *transition model*. Hidden states generate *observations* (data) from which the hidden state is to be inferred. This inference is aided by maintaining a *sensor model*. The discrete time case will be discussed here. Following the notation in §15 of [323], let X_t denote the set of unobservable state variables at time t. Let E_t denote the random variable of observations at time t. Lower case will be used to indicate actual

data value samples from the random variables, for example e_t will denote a fixed set of values of the observation at time t from the random variable E_t. Subscripts separated by a colon, as in $A_{a:b}$ refer to all values of the variable A at all times from a to b, that is $A_a, A_{a+1}, \cdots, A_b$.

The transition model defines how the hidden states change with time by $P(X_t|X_{0:t-1})$. The sensor model $P(E_t|X_{0:t-1}, E_{0:t-1})$ defines the probability for making different observations at time t given information about the previous states and observations. The *Markov assumption* is that the current hidden states depend on only a fixed number of time steps into the past (§15 of [323]). For example, first-order Markov transition and sensor models could be written respectively as:

$$P(X_t|X_{0:t-1}) = P(X_t|X_{t-1}) \tag{2.37}$$

$$P(E_t|X_{0:t-1}, E_{0:t-1}) = P(E_t|X_t) \tag{2.38}$$

The current states of n-th order Markov process would depend on a maximum of n earlier time steps.

Introducing a prior distribution over initial hidden states, $P(X_0)$, in combination with the transition and sensor models allows (assuming a first-order Markov model) the joint distribution over states and observations to be expressed as:

$$P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=1}^{t} P(X_i|X_{i-1}) P(E_i|X_i) \tag{2.39}$$

A graphical model of an illustrative time dependent Bayesian Inference joint distribution, the Hidden Markov Model (HMM), is shown in Figure 2.3. Dynamic Bayesian Networks are a more general formulation of the time dependent inference problem and are described in [323]. The HMM is particular subcase of a Dynamic Bayesian Network, as is Kalman Filtering (see §17 and §18 of [275]).

Following §15.2 in [323], inference using temporal models, as in equation (2.39), can be classified into several different tasks. The main four tasks are Filtering, Prediction, Smoothing and Most Likely Explanation estimation. *Filtering* involves calculating the belief regarding the hidden state given all available evidence (the posterior distribution over X_t), that is filtering is the task of computing $P(X_t|e_{1:t})$. *Prediction* computes beliefs regarding future states,

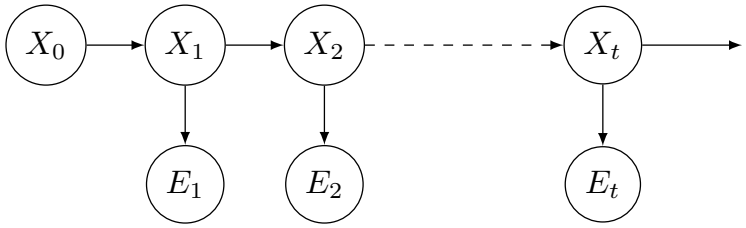


Figure 2.3: Hidden Markov Model Bayesian Network Diagram. Random variables X_0, X_1, X_2 and X_t denote hidden variable at times $0, 1, 2$ and t. Random variables E_1, E_2 and E_t denote observed variable at times $0, 1, 2$ and t.

X_{t+k} for $k > 0$, given evidence available in the present, $e_{1:t}$. That is, prediction computes the posterior $P(X_{t+k}|e_{1:t})$ for $k > 0$ and as such is like filtering without including new data. *Smoothing* uses observational data from the future, $e_{1:t}$, to update beliefs regarding past states, X_k for $0 \leq k < t$. By using more observational data, smoothing can provide better estimates of past behaviour than was available at the time the observation was made. The *Most Likely Explanation* in this context refers to the most likely set of hidden states given the observational data, that is $\text{argmax}_{x_{1:t}} P(x_{1:t}|e_{1:t})$.

To summarise each of these inference tasks are:

$P(X_t\|e_{1:t})$	Filtering
$P(X_{t+k}\|e_{1:t})$ for $k > 0$	Prediction
$P(X_k\|e_{1:t})$ for $0 \leq k < t$	Smoothing
$\underset{x_{1:t}}{\text{argmax}}\ P(x_{1:t}\|e_{1:t})$	Most Likely Explanation

Under a Bayesian Inference framework, the above tasks are carried out by splitting the dependencies between states and observations across time using Bayes Theorem and the sensor and transition models. The temporal dependencies are handled by a recursive formulation that enables for computations to be carried forward and backward in time.

For example, for a first-order Markov model, the recursive posterior updates for single time steps can be written (following the full derivations in §15 of [323]) in terms of messages passed forward and backward through time. The forward messages are used for Filtering and Prediction, whereas the backward messages are used for smoothing. The forward and backward messages are written respectively as:

- Forward message: $f_{1+t} := P(X_t|e_{1:t})$ with $f_{1:0} = P(X_0)$.
- Backward message: $b_{k+1:t} := P(e_{k+1:t}|X_t)$

where the forward and backward messages are computed respectively in terms of functions the forward and backward functions $F(\cdot)$ and $B(\cdot)$. Let $f^f(\cdot)$ denote the filtered forward message and $f^p(\cdot)$ denote the predicted forward message. Then the forward and backward messages are computed for filtering, prediction and smoothing respectively by:

$$f^f_{1:t+1} \propto P(e_{t+1}|X_{t+1})F(f_{1:t}, e_{t+1}) \tag{2.40}$$

$$f^p_{1:t+1} = F(f_{1:t}, e_{t+1}) \tag{2.41}$$

$$b_{k+1:t} = B(b_{k+2:t}, e_{k+1}) \tag{2.42}$$

Note that the forward message update for filtering incorporates new information via the sensor model. Finally, the time-step update functions, $F(\cdot)$ and $B(\cdot)$, can be written:

$$F(f_{1:t}, e_{t+1}) = \sum_{x_t} P(X_{t+1}|x_t)P(x_t|e_{1:t}) \tag{2.43}$$

$$B(b_{k+2:t}, e_{k+1}) = \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})P(x_{k+1}|X_k) \tag{2.44}$$

For discrete models, an efficient smoothing algorithm that uses the above formulation is the forward-backward algorithm (§13 of [275]). Again for discrete models, the Most Likely Explanation can be efficiently estimated by the Viterbi algorithm (§17 of [275]).

With the introduction of temporal structures, numerical problems can be framed in terms of Bayesian Inference. Random or dynamic hidden state estimation problems can clearly by incorporated into the temporal Bayesian Inference framework by following the prescriptions above. Deterministic hidden state estimation problems that rely on iterative algorithms can also be understood as Bayesian Inference problems. Problems such as search and optimisation can be seen as temporal Bayesian Inference by using simple transition models (for example constant transitions from hidden state to hidden state through time). Then, the recursive inference formulation described above can be used to reason about the nature of the estimation problem at hand by treating the observational data as a time ordered sequence. Such an approach

is useful for thinking about the structure of problems in Uncertainty Quantification that, although modelling random phenomena, are themselves deterministic algorithms when run on a classical computer. For example Monte Carlo Sampling (discussed in Section 2.3.4), can be thought of as an inference algorithm with a given sensor model in this way. This is also related to the discussion in [175].

The final task associated with inference in the Bayesian Inference setting is *Learning*. If the transition and sensor models are fully known, then there is no learning required for inference and only state estimates over time need to be calculated. If the transition and sensor models are unknown, then they must be learnt in order to perform accurate inference. In the context of science and engineering, model learning is often conducted by using empirical data to build phenomenological models that approximate behaviour sufficiently well over some reasonable level of detail such that useful predictions of the future behaviour of natural phenomena can be made. These empirical models can be assumed to be approximations to the actual behaviour of nature. Model learning strategies are applied in Chapter 7 to formulate an adaptive basis Element Free Galerkin scheme.

Chapter 3

Probability and Random Fields

Contents

Figures

Chapter 3 Overview

Key developments in Chapter 3 include:

- Section 3.2 discusses background theory relating to the simulation of spatially autocorrelated random fields.
- Section 3.3 presents numerical examples of both Gaussian and non-Gaussian random field simulations.
- Section 3.3.1.4 contributes a method for simulation of random fields with numerically singular covariance matrices.
- Section 3.3.3 contributes a so-called phase field method for simulating non-smooth inclusions in a spatial field.
- Section 3.4 contributes an original, detailed analysis of the computational complexity of random field simulation for Monte Carlo analysis, including an analysis of the numerical accuracy of matrix decomposition methods for random field simulation in Section 3.4.2.

- Section 3.5 provides an overview of techniques from the literature for conditional random field simulation with reference to Bayesian updating of Gaussian processes.

3.1 Introduction

This Chapter details the necessary random field probability theory required to understand the developments in the remainder of this thesis. The goal of Uncertainty Quantification is typically to estimate uncertainty in the outputs of some function given a set of uncertain inputs to the function. Uncertainty Quantification uses probability theory to model the variability potential of the various parameters of some function. Important, relevant references on probabilistic modelling theory that provide additional background detail for this Chapter include [252, 337, 338, 393, 394]. This thesis focusses on Uncertainty Quantification for models of physical phenomena. Models of physical systems for many problems of interest (such as Partial Differential Equation continuum mechanical models) are too complicated to be solved analytically and numerical methods must be used. Further, when considering simulation models of physical phenomena, the geometry over which the physical behaviour is to be analysed must be factored into the probabilistic estimates of parameter uncertainty.

It is often the case that spatially distributed properties of a physical model display variability, but in such a way that nearby values of a parameter field are more similar and distant parts of the parameter field are less similar. For example, the shear strengths at two different points in some soil are more likely to be similar the closer the two points are to one another [116, 117, 298, 159]. This property is referred to as spatial autocorrelation. Other example applications of spatial autocorrelation include modelling of structural material properties [255], diffusion coefficient variability in transport phenomena [100], and electrostatics [230] as well as applications beyond engineering, for example, population genomics [109] and city-wide health data analysis [220]. Random fields can be used to describe and model these features of physical systems using probability theory in a rigorous way [381, 5, 4]. The focus of this Chapter is, however, on the abstract methodological aspects of random field modelling. It is noted here that [158] (published work by the author) includes a physical

example of random field modelling for a Geotechnical slope stability model. The reader interested in a direct application example of the techniques in this Chapter is referred to [158]. Further physical problem based examples are presented in this thesis, in particular in Chapter 4.

This Chapter details relevant aspects random field theory and then describes several methods for random field simulation. Gaussian and non-Gaussian random field modelling methods (based on copula theory [280]) are detailed. A numerical error analysis for a fast random field simulation methodology is detailed. Random field simulation is utilised for physical system Uncertainty Quantification in the subsequent chapters of this thesis. As well as smooth random fields, this Chapter also describes techniques for phase field and threshold random simulation. Modelling of discrete features in a spatially distributed field has applications many applications [229]. Relevant examples for Civil Engineering include modelling fractures in quasi-brittle materials or in simulating aggregate within concrete. Several examples of Gaussian, non-Gaussian and phase field random field model simulations are presented. These field simulation techniques demonstrate the potential range of input parameter distributions that can be modelled and indicate that analytical solutions of many Uncertainty Quantification problems are not feasible. This serves to further motivate the introduction of sampling based Uncertainty Quantification methods detailed in the later Chapters of this thesis. An analysis of the computational complexity of random field methods for use with sampling based Uncertainty Quantification is described, based on the analysis published by the author in [158]. Further, conditional random fields and their application to estimating random field parameters from data are also discussed.

3.2 Random field theory overview

To describe randomly varying functions over space and time, random fields combine both geometry and probability theory. Following on from the mathematical preliminaries and probability theory introduced in Chapter 2, this Section defines further functional analytic and geometric concepts before introducing random fields. In particular, a discussion of the eigenvalues and eigenfunctions of linear operators on function spaces is provided, expanding on the mathematical preliminaries presented in Chapter 1. Gaussian random

fields are introduced. Using the definitions of the eigenvalues and eigenfunctions of operators, the Karhunen-Loève Expansion representation of a Gaussian random field is discussed. Further, non-Gaussian random fields defined via copula theory are also described. This background material introduces the theory necessary to properly understand random field simulation techniques which are described in Section 3.3.

3.2.1 Random fields

A random field is a measurable mapping from a probability space to the space of all functions (of a given particular type) over a topological space. Essentially, in this framework, all functions in a function space over some domain are assigned a probability. Specifically, with reference to Definition 1.1.1 in [5], a random field is defined as:

Definition 3.2.1. *Random Field:* Let $(\Omega, \mathcal{F}, P)$ be a probability space and T a topological space. Let E^T be the space of all E-valued functions on T. A measurable map, f, with:

$$f : \Omega \to E^T \tag{3.1}$$

is called an E^T-valued random field. △

When considering the physical systems of interest in this thesis, manifolds provide a useful characterisation of the spaces of interest [236]. The manifold structure of the random field definition is important as it enables the distance between points to be defined clearly. For practical problems, modelling the time and space similarity of random fields requires a way to define the distance between points. Additionally, there is a close correspondence between manifold metric tensors and correlation structures (for example see §12.2 in [5]). Although these geometric aspects will not be dealt with explicitly in this thesis, it is useful to keep the underlying mathematical structure of random fields in mind.

In the case that the topological space is a manifold, the space of functions of interest is some fibre bundle, E, over the base space M. The random field map takes samples $\omega \in \Omega$ (in a measurable manner as per Definition 2.5.3) to a section e of E such that the probability of the section, $P(e)$, is given by the probability $P(f^{-1}(e))$ where $f^{-1}(e) \in \mathcal{F}$. The most important type of random fields for this thesis will be real valued vector fields over subsets

of N-dimensional Euclidean space, $T \subset \mathbb{R}^N$. A real vector-valued random field on part of N-dimensional Euclidean space, $T \subset \mathbb{R}^N$, would map a set of d-dimensional vectors, $\mathbb{R}^d$, to each point in T.

From §5 in [195], two useful ways of considering random fields are as either a function parameterised by the outcome set of a probability space or as a random function parameterised over the topological space T. As an example, consider a random real valued function over $[0, \infty) \in \mathbb{R}$ where $t \in [0, \infty)$ is interpreted as time. This type of random field is referred to as a *stochastic process* [90]. The random function of interest could be described as:

$$f : [0, \infty) \times \Omega \to \mathbb{R}$$

where Ω is the space of outcomes in a probability space. Then, for $t \in [0, \infty)$ and $\omega \in \Omega$, $f(t, \omega)$ is the value of the process at time t for the outcome ω.

Fixing $\omega \in \Omega$ gives a deterministic function over time:

$$f^\omega : t \mapsto f(t, w) \tag{3.2}$$

which is termed a *realisation* of the process. With this parametrisation, the functions f^ω are a collection of functions for $\omega \in \Omega$ with probability defined by the probability measure $P(\omega)$. As such, the probability measure is a distribution over a function space.

Instead of fixing the random event, the parametrisation over the topological space can be fixed. In the case of this example, this means fixing t. Then f_t is a random variable:

$$f_t : \omega \mapsto f(t, w) \tag{3.3}$$

Viewed in this way, the random field is a collection of random variables indexed by the topological space parametrisation, t. The probability measure, then, describes the joint distribution of these random variables defined at each $t \in T$.

As an example, before considering random fields and their simulation in more detail, consider the Ornstein-Uhlenbeck process, a particular Stochastic Differential Equation, defined in [378, 145] as:

$$dx_t = \theta \left(\mu - x_t \right) + \sigma dW_t \tag{3.4}$$

where $\theta > 0$ is a scaling parameter, μ is the mean and $\sigma > 0$ is the drift parameter. W_t denotes a white noise process, that is, Gaussian noise (also known as a Weiner Process, see §14.1 of [90]). The Ornstein-Uhlenbeck process is a type of so-called mean reverting process in that, despite random fluctuations in the function value, it tends towards a central value over time. Figure 3.1 demonstrates the different parametrisations of a random field shown in equations (3.2) and (3.3) diagrammatically. The realisations of the Ornstein-Uhlenbeck process shown were generated by the Euler-Murayama Method, the details of which are discussed in [213, 60]. Other techniques for simulating random fields are discussed at length in Section 3.3 after the necessary mathematical formalities are provided. The process realisations in Figure 3.1 demonstrate a number of features of random fields of the types of interest in this thesis over real function spaces more generally. In particular, despite random fluctuations, points nearby in time are more similar than points separated by large times. It is this feature random fields that will be of the most interest when considering how the properties of physical systems vary across space and time.

3.2.1.1 Gaussian Random Fields

When considering physical systems, it is often useful to think of random fields as the joint probability distribution over space and/or time of some random function. It is often desirable to specify a pointwise marginal distribution for the random variables of the type in equation (3.3). For example, measurements of soil strengths across some area may be found to follow a beta distribution. In this case, it would be desirable for a random field model of the real physical scenario to replicate this fixed point marginal distribution behaviour. The most common type of random field encountered when simulating physical systems is the Gaussian random field. These random fields use Gaussian random variables to model the pointwise marginal distribution. A Gaussian random variable is defined, following §1.2 of [5] and [224], as follows:

Definition 3.2.2. *Gaussian Random Variable:* A random variable X is said to be *Gaussian* or *normally distributed* if it has the probability density function:

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad x \in \mathbb{R} \tag{3.5}$$

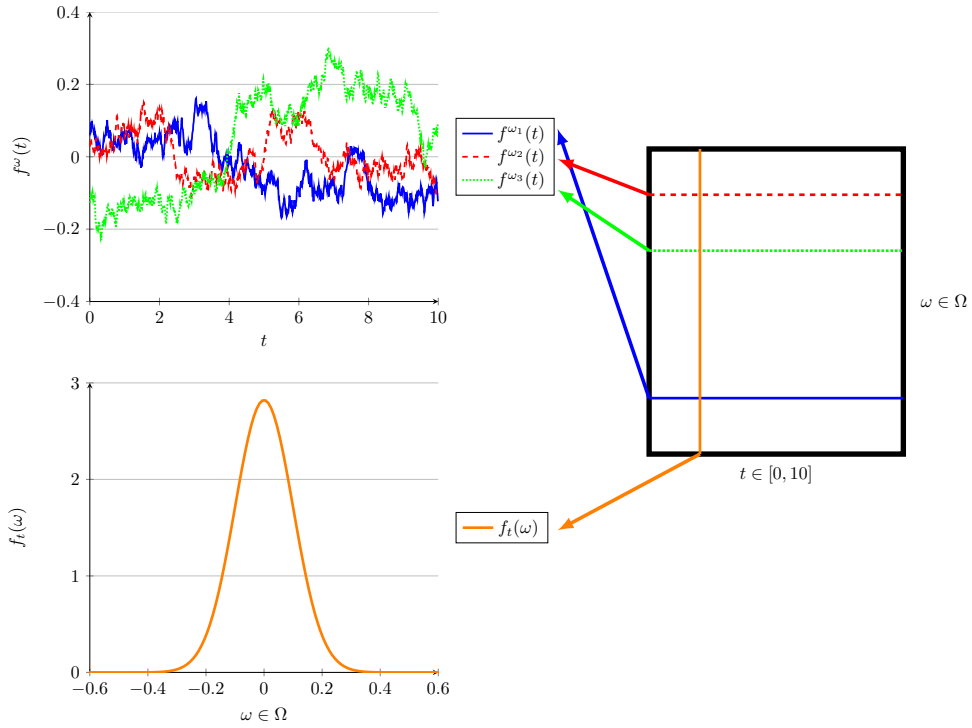


Figure 3.1: A conceptual illustration of two different parametrisations used to model random fields, demonstrated using the Ornstein-Uhlenbeck process. In this case, random fields are also known as random processes or stochastic processes. For the probability space $(\Omega, \mathcal{F}, P)$, each function $f(t, \omega)$ is a random real valued function over the space T (in the diagram, $t \in [0, 10)$) and is assigned a probability $P(\omega)$. The rectangular region on the right of the Figure is a graphical representation of the space $T \times \Omega$. Random processes can be parameterised as per equation (3.2) by considering deterministic functions, $f^\omega(t)$, over T. Three examples of such realisations (or simulations) are shown in the upper left plot. The three fixed ω values are indicated in the rectangle on the right of the Figure by lines with fixed ω as ω_1, ω_2 and ω_3. Conversely, as per equation (3.3), a random process can be parameterised by considering random variables at fixed $t \in T$ by $f_t(\omega)$. The lower left plot indicates the probability distribution for the value of the random real function $f_t(\omega)$ for a fixed t. Where valid, the probability distribution at a fixed t may also be termed the point marginal distribution. The vertical line shown on the illustration of the space $T \times \Omega$ indicates the parameterisation of the $f_t(\omega)$ at a fixed t.

for *mean* μ and *standard deviation* σ. The variance of a Gaussian random variable is given by σ^2. A Gaussian random variable X is also denoted $X \sim \mathcal{N}(\mu, \sigma^2)$. In the case that $\mu = 0$ and $\sigma = 1$, X is said to have a *standard normal distribution.* △

Before introducing the multivariate Gaussian distribution, it is useful to introduce the notion of *positive definite* and *positive semi-definite* matrices:

Definition 3.2.3. *Positive and semi-positive definite matrix:* A matrix $C \in \mathbb{R}^{d\times d}$ is said to be *positive definite* if, for all $x \in R^d$, $x^T Cx > 0$. A matrix $C \in \mathbb{R}^{d\times d}$ is said to be *positive semi-definite* (or *nonnegative definite*) if, for all $x \in R^d$, $x^T Cx > 0$. △

These terms are described in detail in §4 of [148] and §7 of [191].

Following §1.2 in [5] and §3.6 of [224], the multivariate Gaussian distribution extends the Gaussian distribution to vector valued random variables as follows:

Definition 3.2.4. *Multivariate Gaussian distribution:* An $\mathbb{R}^d$-valued random variable X (that is, X is a real valued random vector) is said to have a *multivariate Gaussian distribution* if all linear combinations of the components of $x = [x_1, \cdots x_d] \in \mathbb{R}^d$ are Gaussian random variables. That is, for all $\alpha = [\alpha_1, \cdots \alpha_d] \in \mathbb{R}^d$, the inner product $\langle \alpha, x\rangle = \sum_{i=1}^{d} \alpha_i x_i$ is a Gaussian random variable. Further, given the above conditions, there will exist a *mean vector* $\mu \in \mathbb{R}^d$ with $\mu_i = \mathbb{E}[X_i]$ (the expected value of the $i-th$ component of X) and positive semi-definite *covariance matrix* $\Sigma \in \mathbb{R}^{d\times d}$ with components $\Sigma_{ij} = \mathbb{E}\left[(X_i - \mu_i)(X_j - \mu_j)\right]$. Given these definitions, the probability density of a vector x sampled from the Gaussian random vector X is:

$$g(x) = \frac{1}{(2\pi^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}})} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{3.6}$$

where $|\Sigma|$ is the matrix determinant of Σ. The density of a Gaussian random vector with mean vector μ and covariance matrix Σ is also written $X \sim \mathcal{N}(\mu, \Sigma)$. △

A real valued *Gaussian random field* over T assigns to each point $t \in T$ a random real value, $v(t)$, such that the random variables at each point, t, have a Gaussian distribution. Gaussian random fields have a number of desirable properties. Mathematically, a Gaussian random field, f, is completely described by its first two moments. The first moment is the *mean function* and is given by:

$$\mu(t) = \mathbb{E}\left[f(t)\right] \tag{3.7}$$

The second moment is the *covariance function* which (for $s, t \in T$) is given

by:

$$C(s,t) = \mathbb{E}\left[(f(s) - \mu(s))(f(t) - \mu(t))\right] \tag{3.8}$$

where $C(s,t) : T \times T \to \mathbb{R}$ is also known as a *kernel function*. More details regarding the mean and covariance functions are given in §1.2 of [5] and [381]. Further, from §1.3 in [5], note that Gaussian random fields can be defined to take values over Banach spaces (see [221]) by replacing the correlation function with families of operators from the topological dual to the base space. This level of detail will, however, not be required for the later developments in this thesis.

There are a large number of possible correlation functions, also known as kernel functions, that satisfy the positivity requirements (see [359, 381, 3] and §14 of [275]). Two point correlation functions, $C(s,t)$, of interest for modelling spatial data include:

$$C(s,t) = \frac{-\|s-t\|_2^2}{\theta^2} \qquad \text{Exponential Decay} \tag{3.9}$$

$$C(s,t) = \max(1.0 - \|s-t\|_1, 0) \qquad \text{Linear} \tag{3.10}$$

$$C(s,t) = \frac{1}{(a^2+\tau^2)^\nu} \qquad \text{Rational quadratic} \tag{3.11}$$

As a Gaussian random field depends only on its first and second moments, both sampling and parameter estimation from data is much easier than in the fully general case. The particular nature of the correlation structure between the random variables renders Gaussian random fields amenable to simulation (sampling). In later parts of this thesis, Monte Carlo Simulation is used for Uncertainty Quantification. This requires repeated sampling from random fields and the ability to rapidly sample from complicated random fields facilitates this. Gaussian random field sampling given mean and correlation functions is described later in this Chapter. The Gaussian distribution is, however, not a suitable random model for many parameters one may wish to model probabilistically for Uncertainty Quantification. For example, the Young's Modulus of a material is a non-negative quantity. Using copula theory (described in Section 3.2.1.3), the Gaussian random field correlation structure can be extended to models with non-Gaussian pointwise marginal distributions. This requires only minor extensions to techniques for Gaussian random field simula-

tion. Further, when it is necessary to estimate the spatial correlation structure (that is the covariance function) from real sampling data, it is rarely the case that there will be enough data to fit a very complicated correlation model. The Gaussian correlation structure is sufficiently complicated for many typical Civil Engineering applications. These issues are discussed in [118].

3.2.1.2 Mercer's Theorem and the Karhunen-Loève Expansion

The covariance structure of Gaussian random fields allows them to be represented in terms of a mathematically convenient infinite series expansion. Simulation and discretisation for the random field of interest is based on truncations of this infinite series. The mathematical details are given in this Section in preparation for the details of random field simulation presented in Section 3.3. Briefly, for Gaussian copula, the positive semi-definite covariance function can be seen to be a *Mercer Kernel* (§18 of [323]). Following from this, an infinite series expansion (the *Karhunen-Loève Expansion*), can be used to express a random field in terms of the eigenvalues and eigenvectors of the random field correlation structure.

Following from §3 in [5], a zero mean Gaussian random field, $f(t)$, can be represented in terms of orthogonal basis functions, $\phi(t)$, as

$$f(t) = \sum_{i=1}^{\infty} \xi_i \phi_i(t) \tag{3.12}$$

where each ξ_i is a $\mathcal{N}(0,1)$ random variable and the functions $\phi_i(t)$ are functions on T which are related (via Mercer's Theorem discussed below) to the covariance function of the random field. This representation is known as the Karhunen-Loève Expansion (KL Expansion). Although not considered in detail in this thesis, there are a number of requirements regarding the $\mathcal{L}^2$ convergence of the series expansion representation of a Gaussian random field. These issues are discussed in [5]. Formal statements and proofs of the validity of the series expansion representation of a Gaussian random field are given in §3 of [5] and [4]. Calculating the functions $\phi_i(t)$ is typically done by solving an eigenvalue decomposition problem.

For simulation purposes, it is of more interest to consider how the eigenfunction structure of the correlation function can actually be calculated. Assume that

the Gaussian random fields of interest are taken over a compact subset, T of $\mathbb{R}^N$ (that is, $T \subset \mathbb{R}^N$), and that the field is square integrable as defined in Chapter 1 (in other words, $f(t) \in \mathcal{L}^2$ space). Further, assume that the Gaussian random field of interest has mean zero so that $E[f(t)] = \mu(t) = 0$. Consider the linear integral operator $T_C : \mathcal{L}^2(T) \to \mathcal{L}^2(T)$ (see §8.1.4 of [310]) given by:

$$(T_C\phi)(t) := \int_T C(s,t)\phi(s)ds \tag{3.13}$$

so that $T_C : T \times T \to \mathbb{R}$ is, specifically, a *Hilbert-Schmidt Kernel* (see Lemma 8.20 in [310]) meaning that:

$$\int_T \int_T |C(s,t)|^2 dsdt < \infty \tag{3.14}$$

where T_C is also known as a *Hilbert-Schmidt Kernel Integral Operator*.

Let $\lambda_1, \lambda_2, \cdots$ be the eigenvalues (ordered such that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots$) with associated eigenfunctions $\phi_1, \phi_2, \cdots$ of the integral operator T_C. Specifically, the eigenfunctions and eigenvalues refer to the solutions of the integral equation:

$$(T_C\phi_i)(t) = \int_T C(s,t)\phi_i(s)ds = \lambda_i\phi_i(t) \tag{3.15}$$

Additionally, the eigenfunctions are normalised to be orthogonal such that the inner product of the eigenfunctions is the Kroenecker delta:

$$\langle \phi_i, \phi_j \rangle = \int_T \phi_i(t)\phi_j(t)dt = \delta_{ij} \tag{3.16}$$

The theory of the eigenvalues and eigenfunctions of integral operators is known as Fredholm Theory and is discussed at length in [87].

From Theorem 3.2.1 in [5](due to Mercer), a positive semi-definite kernel function admits a decomposition of the form:

$$C(s,t) = \sum_{i=1}^{\infty} \lambda_i\phi_i(s)\phi_i(t) \tag{3.17}$$

With these definitions in place, the KL Expansion theorem can be stated, following [141], §2.4.3 of [356] and §3.2 of [5], as follows:

Theorem 3.2.1. *Karhunen-Loève Expansion:* Let $f(t)$ be a zero-mean square integrable random field on $(\Omega, \mathcal{F}, P)$ over $T \subset \mathbb{R}^N$ with covariance

function $C(s,t)$ (a Mercer kernel). Let the operator T_C, see equation (3.13), have eigenfunctions $\{\phi_i(t)\}_{i=1}^n$ that form an orthonormal basis on $\mathcal{L}^2(T)$ with associated eigenvalues $\{\lambda_i\}_{i=1}^n$ (ordered such that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots$). Then $f(t)$ has the representation:

$$f(t) = \sum_{i=1}^{\infty} f_i \phi_t(t) \tag{3.18}$$

with convergence of the summation in $\mathcal{L}^2$ that is uniform in t with:

$$f_i = \int_T f(t)\phi_i(t)dt \tag{3.19}$$

Further, the f_i are independent random variables with zero mean variance λ_i such that:

$$\mathbb{E}[f_i] = 0$$
$$\mathbb{E}[f_i \cdot f_j] = \lambda_j \delta_{ij}$$

for all integers i and j.

Proof. (Proof sketch) Begin by projecting the process $f(t)$ onto the eigenvalue basis, $\{\phi_i(t)\}_{i=1}^{\infty}$, using the $\mathcal{L}^2$ inner product by:

$$f(t) = \sum_{i=1}^{\infty} \langle f(t), \phi_i(t) \rangle \phi_i(t) \tag{3.20}$$

where

$$f_i := \langle f(t), \phi_i(t) \rangle = \int_T f(t)\phi_i(t)dt \tag{3.21}$$

Each f_i is a random variable.

Then the expectation of the each f_i coefficients from the eigenfunction basis

projection can be expressed as:

$$\begin{aligned}\mathbb{E}[f_i] &= \mathbb{E}\left[\int_T f(t)\phi_i(t)dt\right] \\ &= \int_T \mathbb{E}\left[f(t)\right]\phi_i(t)dt \\ &= \int_T 0 \cdot \phi_i(t)dt \\ &= 0\end{aligned}$$

Further, the covariance of the random variable coefficients of the eigenvalue projection can be expressed as:

$$\begin{aligned}\mathbb{E}[f_i \cdot f_j] &= E\left[\int_T f(t)\phi_i(t)dt \int_T f(s)\phi_j(s)ds\right] \\ &= \mathbb{E}\left[\int_T \int_T f(t)f(s)\phi_i(t)\phi_j(s)dtds\right] \\ &= \int_T \int_T \mathbb{E}\left[f(t)f(s)\right]\phi_i(t)\phi_j(s)dtds \\ &= \int_T \int_T C(s,t)\phi_i(t)\phi_j(s)dtds \\ &= \int_T \phi_j(s)\left(\int_T C(s,t)\phi_i(t)dt\right)ds\end{aligned}$$

using equation (3.15), the term $\left(\int_T C(s,t)\phi_i(t)dt\right)$ is equal to $\lambda_i\phi_i(s)$ so:

$$\begin{aligned}\mathbb{E}[f_i \cdot f_j] &= \int_T \phi_j(s)\lambda_i\phi_i(s)ds \\ &= \lambda_i \int_T \phi_j(s)\phi_i(t)ds \\ &= \lambda_i\langle\phi_j(s), \phi_i(s)\rangle \\ &= \lambda_i\delta_{ji}\end{aligned}$$

Showing that the summation representation of the process converges in $\mathcal{L}^2$ uses Mercer's Theorem. Truncating the summation to m terms yields:

$$f_m(t) = \sum_{i=1}^{m} f_i\phi_i(t) \tag{3.22}$$

Expanding the expectation of the truncated sum squared error, $\mathbb{E}\left[|f(t) - f_m(t)|^2\right]$,

yields:

$$
\begin{aligned}
&\mathbb{E}\left[|f(t) - f_m(t)|^2\right] \\
&\quad = \mathbb{E}\left[f(t)^2\right] - 2\mathbb{E}\left[f(t) \cdot f_m(t)\right] + \mathbb{E}\left[f_m(t)^2\right] \\
&\quad = C(t,t) - 2\mathbb{E}\left[f(t) \cdot \sum_{i=1}^{m} f_i \phi_i(t)\right] + \mathbb{E}\left[\sum_{i=1}^{m} f_i \phi_i(t) \sum_{j=1}^{m} f_j \phi_j(t)\right] \\
&\quad = C(t,t) - 2\mathbb{E}\left[f(t) \cdot \sum_{i=1}^{m} \langle f(t), \phi_i(t) \rangle \phi_i(t)\right] + \sum_{i=1}^{m}\sum_{j=1}^{m} \mathbb{E}\left[f_i f_j \phi_i(t) \phi_j(t)\right] \\
&\quad = C(t,t) - 2\mathbb{E}\left[f(t) \cdot \sum_{i=1}^{m} \left(\int_T f(s)\phi_i(s)ds\right) \phi_i(t)\right] + \sum_{i=1}^{m}\sum_{j=1}^{m} \mathbb{E}\left[f_i f_j\right] \phi_i(t)\phi_j(t) \\
&\quad = C(t,t) - 2\sum_{i=1}^{m} \mathbb{E}\left[\left(\int_T f(t)f(s)\phi_i(s)ds\right) \phi_i(t)\right] + \sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \delta_{ji} \phi_i(t)\phi_j(t) \\
&\quad = C(t,t) - 2\sum_{i=1}^{m} \left(\int_T \mathbb{E}\left[f(t)f(s)\right] \phi_i(s)ds\right) \phi_i(t) + \sum_{i=1}^{m} \lambda_i \phi_i(t)\phi_i(t) \\
&\quad = C(t,t) - 2\sum_{i=1}^{m} \left(\int_T C(t,s)\phi_i(s)ds\right) \phi_i(t) + \sum_{i=1}^{m} \lambda_i \phi_i(t)\phi_i(t) \\
&\quad = C(t,t) - 2\sum_{i=1}^{m} \lambda_i \phi_i(t)\phi_i(t) + \sum_{i=1}^{m} \lambda_i \phi_i(t)\phi_i(t) \\
&\quad = C(t,t) - \sum_{i=1}^{m} \lambda_i \phi_i(t)\phi_i(t)
\end{aligned}
$$

then, by Mercer's Theorem, $\sum_{i=1}^{m} \lambda_i \phi_i(t)\phi_i(t)$ converges to $C(t,t)$ as m goes to ∞ then:

$$\mathbb{E}\left[|f(t) - f_m(t)|^2\right] \to 0 \; m \to \infty \tag{3.23}$$

□

For a detailed proof of the theorem see §3.2 of [5].

From §2.4.3 of [356], the Karhunen-Loève Expansion can be used to represent a Gaussian random field. Given a mean function $\mu(t)$, a Gaussian random field $f(t)$ can be written:

$$f(t) = \mu(t) + \sum_{i=1}^{\infty} \xi_i \sqrt{\lambda_i} \phi_i(t) \tag{3.24}$$

where λ_i and $\phi_i(t)$ are the eigenvalues and eigenfunctions as defined in The-

orem 3.2.1. The random variables ξ_i are $\mathcal{N}(0,1)$ random variables. In terms of the KL Expansion, the correlated Gaussian process $f(t)$ can be understood as being generated from a white noise source, ξ_i, which is then scaled to the correct standard deviation, $\sqrt{\lambda_i}$. Finally, the correlation structure of the covariance function kernel, $C(s,t)$, is enforced by the basis eigenfunctions. As shown in the proof sketch of Theorem 3.2.1, the infinite summation in equation (3.23) converges to the correct value as $i \to \infty$. The summation can, therefore, be truncated to yield an approximation to the true Gaussian random field. Gaussian random field simulation theory is discussed in Section 3.3.

3.2.1.3 Non-Gaussian Random Fields by copula theory

Simulating fully arbitrary non-Gaussian random fields is a significant challenge, as discussed in [93, 58, 401, 324, 296]. However, for a process with a second order correlation structure and fixed pointwise marginal distribution, *copula* theory can be used to simulate non-Gaussian fields. When modelling the types of random fields of interest in Civil Engineering, pointwise marginals with second order spatial correlation is typically sufficient as any sort of more complicated model is likely to be unjustified given the limited data available. For example, [117] and [21] demonstrate geotechnical applications of random field models for which data is expensive to obtain. The more complicated the model, the more difficult it is to fit [323, 275, 46] (which is a consequence of information theory [335]). As such, this thesis will describe copula models of non-Gaussian $\mathcal{L}^2$ random fields. Further reading in this area is presented in [356, 136].

Copula theory is described in great detail in [280]. From §2 of [280], expanding Definition 2.10.6 of [280] and with reference to §4.2.2.3 of [356] and §1 of [322], a copula can be defined as:

Definition 3.2.5. *Copula:* A copula, C, is a function over the unit hypercube $[0,1]^N \in \mathbb{R}^N$, with:

$$C : [0,1]^N \longrightarrow [0,1] \tag{3.25}$$

such that C is a joint cumulative distribution function of an N-dimensional random vector with uniform marginal distributions. $\triangle$

Specifically, let $X = (X_1, \cdots, X_N)$ be a random vector with distribution func-

tion $F(x_1, \cdots, x_n) = P(X_1 \leq x_1, \cdots, X_N \leq x_N)$ and marginal distribution functions $F_i, X_i \sim F_i$ for $1 \leq i \leq N$, that is, $F_i(x_i) = P(X_i \leq x)$. From Sklar's Theorem (see [341] and §4.2.2.3 of [356]), a continuous joint cumulative distribution function $F(x_1, \cdots, x_N)$ has a unique representation in terms of the marginal distributions $F_i(x_1)$ and the copula function $C(u_1, \cdots, u_N)$ as:

$$F(x_1, \cdots, x_n) = C(F_1(x_1), \cdots, F_N(x_N)) \tag{3.26}$$

To illustrate the meaning of this representation, first note that (via the probability integral transform, see [11]) that the random variable Y given by $Y = F(X)$ for cumulative distribution function F and random variable X has a uniform distribution. Then consider the random vector given by:

$$(F_1(X_1), \cdots, F_N(X_N)) = (U_1, \cdots, U_N) \tag{3.27}$$

that is, the vector $(F_1(X_1), \cdots, F_N(X_N))$ is a vector of uniform random variables $(U_1, \cdots, U_N)$. The copula of $(X_1, \cdots, X_N)$ is given by:

$$C(u_1, \cdots, u_n) = P(u_1 \leq U_1, \cdots, u_N \leq U_N) \tag{3.28}$$

so that the copula $C(u_1, \cdots, u_n)$ is the joint cumulative distribution for the random vector $(F_1(X_1), \cdots, F_N(X_N)) = (U_1, \cdots, U_N)$. The copula contains information regarding the correlation structure between the different random variables $X_1, \cdots, X_N$.

From §4.2.2.3 of [356], the copula density function is given by the derivative of C as:

$$c(u_1, \cdots, u_N) = \frac{\partial^N C(u_1, \cdots, u_N)}{\partial u_1 \cdots \partial u_N} \tag{3.29}$$

then the probability density $f(x_1, \cdots, f_N)$ corresponding to the joint cumulative distribution $F(x_1, \cdots, x_N)$ is:

$$f(x_1, \cdots, x_N) = c(F_1(x_1), \cdots, F_N(x_N)) f_1(x_1) \times \cdots \times f_N(x_N) \tag{3.30}$$

where $f_i(x_i)$ are the probability density functions corresponding to the marginals $F_i(x_i)$.

For simulation, the definition of a copula can be exploited in order to sample from non-Gaussian random fields. First note that for a uniform distribution,

$U = F(X)$ that $F^{-1}(U) = X$ for a continuous cumulative distribution function F. Then note that:

$$(F_1(X_1), \cdots, F_N(X_N)) = (U_1, \cdots, U_N) \tag{3.31}$$

$$(X_1, \cdots, X_N) = (F_1^{-1}(U_1), \cdots, F_N^{-1}(U_N)) \tag{3.32}$$

So that, given a sample $(u_1, \cdots, u_N \sim C(u_1, \cdots, u_N)$ from the copula, samples from the multivariate distribution of $(X_1, \cdots, X_N)$ can be generated by:

$$(x_1, \cdots, x_N) \sim (F_1^{-1}(U_1), \cdots, F_N^{-1}(U_N)) \tag{3.33}$$

$$(x_1, \cdots, x_N) = (F_1^{-1}(u_1), \cdots, F_N^{-1}(u_N)) \tag{3.34}$$

The most relevant copula function when considering spatially distributed data in Civil Engineering applications is the Gaussian copula (see [301]):

$$C(u_1, \cdots, u_N|\Sigma) := \Phi_\Sigma^N(\Phi^{-1}(u_1), \cdots, \Phi^{-1}(u_n)) \tag{3.35}$$

where $\Phi^{-1}(u_i)$ is the inverse standard normal cumulative distribution function and Φ_Σ^N is the N-dimensional normal cumulative joint distribution function with mean zero and covariance matrix Σ.

The Gaussian copula density is given by:

$$c(u_1, \cdots, u_N|\Sigma) := \frac{1}{|\Sigma|^{\frac{1}{2}}} \exp\left(\Phi^{-1}(u)^T \left(\Sigma^{-1} - \mathbf{I}\right) \Phi^{-1}(u)\right) \tag{3.36}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix, $|\Sigma|$ is the determinant of Σ and $\Phi^{-1}(u)$ is the vector:

$$\Phi^{-1}(u) = \begin{bmatrix} \Phi^{-1}(u_1) \\ \vdots \\ \Phi^{-1}(u_N) \end{bmatrix} \tag{3.37}$$

As will be expanded in Section 3.3.2, given a technique for simulating a Gaussian random field, a non-Gaussian random field with a Gaussian copula correlation structure can be simulated using the above transforms. This is the main motivation for introducing copula theory in this thesis.

3.3 Random Field Simulation

For Uncertainty Quantification of physical systems it is often necessary to generate realisations of samples of the random field. Monte Carlo Simulation, for instance, requires repeated sampling from a given distribution in order to calculate the expectation values. Series Expansion methods for Stochastic PDE may also require realisations of a random field to be generated as a type of integral quadrature point [356, 351]. When numerically simulating random fields over spaces with an infinite number of points such as N-dimensional Euclidean space, $\mathbb{R}^N$, some form of discretisation must be introduced to enable a finite length representation of the field simulation. This Section will discuss the discretisation and simulation of random fields.

Gaussian and non-Gaussian random field simulation over subsets of Euclidean space will be introduced. Modelling spatial autocorrelation for the types of fields of interest in this thesis can, to a sufficient level of detail, be carried out using Gaussian copula correlation structures. More general expositions regarding spatially autocorrelated non-Gaussian random fields are given in [93, 324]. This Section first details techniques for simulating Gaussian random fields. Then, using Gaussian random field techniques, simulation methodologies for non-Gaussian fields with Gaussian copula correlation structures is described. As these non-Gaussian random fields can be generated by transformations from Gaussian random fields, Gaussian random field simulation also enables this form of non-Gaussian random simulation. This Section also describes a method for generating novel random field samples by combining random fields with point processes and phase field models. This is demonstrated by the phase field simulation of a circular inclusions in a random field and phase field simulation of jointed rock mass. A number of example random field simulations are presented in Figures 3.2, 3.3, 3.5, 3.6 and 3.7. All random field simulations were generated using the Python SciPy package [204].

Simulation and discretisation of random fields in this Section is based on a truncated orthogonal function series expansion. The mathematical details are given in this Section. Briefly, for Gaussian copula, the positive semi-definite covariance function can be seen to be a Mercer Kernel, as discussed in Section 3.2. Following from this, an infinite series expansion (the KL Expansion), can be used to express a random field in terms of the eigenvalues and eigenvec-

tors of the correlation function. Truncating the infinite series expansion to a fixed number of eigenbasis functions allows for a discrete representation of a random field. For a Gaussian random field, the eigenbasis decomposition can be combined with a vector of independent Gaussian samples to generate a random field sample. To generate a non-Gaussian field with a Gaussian copula and an alternative pointwise marginal distribution, it is possible to generate a Gaussian random field, recover the copula distribution and then apply the desired marginal function to generate the final random field simulation. This Section details methods of covariance matrix decomposition based simulation for Gaussian random fields. As well as eigenvalue and Cholesky decomposition methods, an analysis of a Cholesky decomposition based method suitable for dealing with numerically singular covariance matrices is presented, based on work published by the author in [158]. The numerical examples in [158] also demonstrate the random field simulation techniques described in this Chapter for a Geotechnical Engineering problem. In particular, probabilistic simulations of Mohr-Coulomb soil constitutive model parameters are presented. Further physical examples of the techniques demonstrated in the following sections are given in the following Chapters of this thesis.

3.3.1 Gaussian Random Field Simulation by Covariance Matrix Decomposition

Following the discussion in Section 3.2.1.2 on the KL Expansion, a Gaussian random field can be represented by truncating an infinite series expansion, as in equation (3.22). In practical terms, this means that the covariance structure of a random field can be approximated at N points as a matrix with entries C_{ij} given by the covariance function evaluated between points x_i, x_j for $i, j \leq N$. Using the "correlating" properties of the eigenvectors of the covariance matrix, a vector of standard normal samples can be converted to a correlated sample from a (truncated approximation) to a correlated zero mean Gaussian random field. A specified mean vector with constant valued entries can be added on to the correlated sample vector as needed. Simulations of Gaussian random fields using decompositions of the covariance matrix are referred to as covariance matrix decomposition methods. Other methods are discussed below, however, the focus of this thesis will be restricted to in depth discussions of covariance matrix methods.

The majority of the simulation based Uncertainty Quantification analyses in this thesis use finite element basis functions to discretise fields over spatial regions. When simulating random fields for subsequent mapping onto the basis used for PDE analysis, it is not necessary to use the same discretisation for the random field as that used for the PDE numerical analysis. However, using the same basis functions for all fields is the most convenient for simulation. Covariance matrix decomposition simulation of Gaussian random fields, using the PDE analysis basis functions, will be used for the analyses later in this thesis and is discussed in this Section. In this Section, a justification for the use of matrix decomposition methods is presented in terms of computational complexity and numerical precision. Methods for resolving numerical stability problems that may arise with matrix decomposition random field generators are also discussed.

While several methods exist for the simulation of random fields, as detailed in [118], the majority of these methods are only appropriate for generating random field samples discretised by rectangular, or almost rectangular, grids. For example, LAS and the fast Fourier transform (FFT) algorithms generate random fields over regular grids. Complicated mesh geometry can destroy the required form of the random field correlation structure that make LAS and FFT generators efficient and would be difficult to apply to arbitrary basis functions sets. One possible technique to do so would be to generate a random field over a sufficiently fine regular grid and then project this fine field onto the desired basis functions. Such a method would prevent the correlation structure from being discretised accurately unless a very fine grid was used. It may also be possible to use some curvilinear mapping of, say, arbitrarily shaped finite element mesh basis elements to some regular rectangular grid. FFT or LAS could then be applied to this rectangular grid and then an inverse transform applied. Such a transformation from the unstructured space to the rectangular space and its inverse is would not, in general, be bijective. An arbitrary basis formulation of LAS, potentially based on renormalisation group flow (see [385]), would be an interesting topic for future research.

Alternatively, the Turning Bands Method (TBM), detailed in [118], is suitable for handling arbitrary geometry. TBM involves generating a series of one dimensional random field samples along arbitrarily oriented lines and building the desired higher dimensional random field sample from these line processes.

At each point in the discrete output field, a weighted contribution from each one dimensional random field is added based on the perpendicular distance of the point to the line. However, when the correlation function is anisotropic, TBM can be difficult to use as discussed in [115]. The precision of the method is dependent on both the one dimensional random field generator and the number of individual lines used. If an insufficient number of lines are used, streaked patterns emerge that may cause unrealistic preferential stress or flow paths during subsequent PDE analysis [115, 118]. Finally, difficulties also arise in deciding how to discretise the one dimensional line processes and the interaction of this discretisation with arbitrary basis functions. If, for example, a very dense finite element discretisation is used to minimise errors caused by the generation of the line process, any potential efficiency gains from using a graded mesh will be lost. Alternatively, some discretisation that is calculated based on the relative orientation and size of finite element basis mesh elements to the current line process could be calculated although this calculation would be, again, necessarily constrained by the smallest elements in the mesh and difficult to calculate. Once the discretisation of the line processes becomes fine enough, then essentially one is generating a random field over a fine grid and then mapping this by projection to some other set of basis functions. The problems with this are discussed above.

Methods based on decomposition of the covariance matrix are able to generate random fields with an accurate correlation structure over arbitrary basis functions with accuracy defined by the expansion order of the KL expansion. The structure of the basis functions are directly incorporated into the simulation, without any need for remapping the random field to the PDE analysis mesh. Matrix decomposition methods include the Cholesky, LDL and Modal decompositions [148, 224, 118]. Very roughly, these methods extract a set of basis vectors from the covariance matrix, such that a set of uncorrelated random samples can be transformed into a correctly correlated sample from the random field. For a Gaussian copula, pairs of basis vectors can be considered as representing an ellipse. The correlation between a pair of basis vectors is effectively represented by the orientation of this ellipse. Techniques for random field generation by covariance matrix decomposition and potential pitfalls are discussed in the remainder of this Section.

Matrix decomposition, in particular the Cholesky decomposition, is widely

used in many fields to generate samples from multivariate normal distributions [224]. The theoretical properties of matrix decomposition techniques in terms of infinite precision arithmetic are first addressed. Any practical implementation will be limited to some finite precision approximate calculation. The implications of finite precision floating point arithmetic on the decomposition of the covariance matrix are discussion in Section 3.4.2.

All matrix decomposition random field generators first start by forming the covariance matrix $\mathbf{C}$. The covariance matrix is always symmetric [118] and positive semi-definite [191]. These properties will be relevant for the rest of this Section. For a locally averaged random field (see [158, 118]) over finite element-type locally supported basis functions, the entries of $\mathbf{C}$ are the covariances between local averages. For n elements in a finite element mesh, $\mathbf{C}$ is of size $n \times n$. The entries of $\mathbf{C}$ are given by C_{ij} equal to $\mathrm{Cov}\left[X_i, X_j\right]$.

3.3.1.1 Eigenvalues and eigenvectors of the covariance matrix

To facilitate discussion of random field sampling by covariance matrix decomposition, it is useful to first explore the eigenvalues and eigenvectors of the covariance matrix. By Theorem 8.1.1 in [148], given a symmetric matrix $\mathbf{C}$ of size $n \times n$, there exists an orthogonal matrix $\mathbf{Q}$ (that is, all column vectors are orthogonal) such that:

$$\mathbf{Q}^T\mathbf{C}\mathbf{Q} = \mathbf{\Lambda} = \mathrm{diag}\left(\lambda_1, \ldots, \lambda_n\right) \tag{3.38}$$

Where T denotes the matrix transpose. Also, $\mathrm{diag}\left(\lambda_1, \ldots, \lambda_n\right)$ refers to a diagonal matrix, a matrix with zero in all entries except along the main diagonal which instead has λ_1 in the first row, λ_2 in the second row and so on. Equation (3.38) can be reorganised to the form:

$$\mathbf{C}\mathbf{Q} = \mathbf{Q}\mathbf{\Lambda} \tag{3.39}$$

Presented in this way, $\mathbf{Q}$ is more clearly the matrix whose columns are the eigenvectors of $\mathbf{C}$ and $\mathbf{\Lambda}$ stores the corresponding eigenvalues along the main diagonal. As the eigenvalues can always be reordered into any permutation, let λ_i denote the $i-th$ largest eigenvalue of $\mathbf{C}$. As $\mathbf{C}$ is positive semi-definite, all of the eigenvalues, $\lambda_1, \ldots, \lambda_n$ are greater than or equal to zero (Theorem

4.1.10 in [191]). Then, if there are k non-zero eigenvalues:

$$0 = \lambda_n = \cdots = \lambda_{k+1} < \lambda_k \leq \cdots \leq \lambda_1 \tag{3.40}$$

By Theorem 4.1.10 of [191], if all of the eigenvalues are strictly greater than zero, that is $k = n$ in equation (3.40), then $\mathbf{C}$ is positive definite (rather than semi-definite). By Observation 1.1.7 in [191], a matrix is singular if and only if 0 is in the set of eigenvalues. By conditions 3.7.5 and 3.7.6 (and the proof of their equivalence) in [264], if an $n \times n$ matrix is non-singular then the rank (number of linearly independent columns) of the matrix is n. So then a symmetric positive definite matrix of size $n \times n$ is non-singular and has full rank (all columns are linearly independent).

Conversely, the rank-nullity theorem states that for an $n \times n$ matrix $\mathbf{C}$:

$$n = \text{rank}(\mathbf{C}) + \dim(\text{nullspace}(\mathbf{C})) \tag{3.41}$$

where nullspace($\mathbf{C}$) is defined by $\mathbf{C}x = 0$ where x is a vector of length n. But if zero is in the set of eigenvalues, the matrix is singular and so rank($\mathbf{C}$) $< n$ and the columns of $\mathbf{C}$ have linear dependencies. From [129]:

$$\dim(\text{nullspace}(\mathbf{C})) = k \tag{3.42}$$

Where k is the number of zero eigenvalues as in equation (3.40).

Combining equation (3.41) and (3.42) the results concerning singularity and positive definiteness can be summarised. Let k be the number of eigenvalues equal to 0 of $\mathbf{C}$. If k is greater than zero, then $\mathbf{C}$ is singular and positive semi-definite. If $k = 0$, then $\mathbf{C}$ is positive definite. Additionally, the number of linearly dependent columns of $\mathbf{C}$ is equal to k so:

$$\text{rank}(\mathbf{C}) = n - k \tag{3.43}$$

It is useful to provide a bound on the eigenvalues of $\mathbf{C}$ to facilitate the rounding error analysis in Section 3.4.2. As the effect of variance reduction on σ only reduces the order of magnitude of σ, the effect of γ on σ can be excluded to simplify the eigenvalue bound estimate without changing the rounding error analysis. For a stationary random field, σ is constant across the field. Then a

scalar factor of σ^2 can be taken from $\mathbf{C}$ so:

$$\mathbf{C} = \sigma^2 \mathbf{R} \tag{3.44}$$

where $\mathbf{R}$ is the correlation matrix with entries bounded between -1 and 1.

From equation (3.39), $\mathbf{CQ} = \mathbf{Q\Lambda} = \sigma^2 \mathbf{RQ}$ so:

$$\mathbf{RQ} = \sigma^{-2} \mathbf{Q\Lambda} \tag{3.45}$$

Then, with reference to equation (3.39), the eigenvalues of $\mathbf{R}$ are proportional to $\sigma^{-2}\mathbf{\Lambda}$. From the discussion in Section 3.3.1.2, taking the eigenvalue decomposition of $\mathbf{C}$ gives eigenvalues proportional to σ^2. Then the eigenvalues of $\mathbf{R}$ are proportional to unity. From this the largest eigenvalue, $\lambda_{max}(\mathbf{C})$, for a stationary random field is:

$$\lambda_{max}(\mathbf{C}) \lesssim \sigma^2 \tag{3.46}$$

3.3.1.2 Random field simulation by eigenvalue decomposition for positive definite covariance matrices

It is instructive to examine the generation of random fields first by eigenvalue decomposition (essentially discretised KL Expansion). This decomposition is one of several possible decompositions of the covariance matrix that can be used to generate random fields. First, the simpler case of a positive definite matrix is considered. In infinite precision arithmetic, a positive definite covariance matrix may occur, for example, when the covariance function is monotonically decreasing. By Definition 6.1.9 in [191], a matrix is strictly diagonally dominant if the diagonal entries are strictly greater than all other entries in the matrix. By Theorem 6.1.10, Conditions a and b of [191], a strictly diagonally dominant matrix is non-singular and has positive real eigenvalues. If the covariance function is monotonically decreasing (for example, if an exponential decay correlation function is used), the diagonal entries in the covariance matrix are always greater than the diagonals as the largest correlation occurs between an element and itself.

Even in the case of a monotonically decreasing correlation function, finite

precision arithmetic may result in the covariance matrix becoming diagonally dominant, rather than strictly diagonally dominant. Then the diagonal entries are greater than or equal to the rest of the entries in the matrix. Indeed, if care is not taken with algorithms, diagonal dominance can be lost completely. Numerical stability issues are addressed in Section 3.4.2.

Given an $n \times n$ positive definite covariance matrix decomposed into eigenvectors and eigenvalues as $\mathbf{C} = \mathbf{Q}\mathbf{\Lambda}$, a sample from Gaussian random field, $Z(x, y)$, can be generated:

$$Z(x, y) = \mathbf{Q}\mathbf{\Lambda}^{(1/2)}a + \mu \tag{3.47}$$

where a is a vector of size n whose entries are given by random samples of a standard normal distribution and μ is the mean of the random field [390, 118].

Equation (3.47) can be understood as generating a random field in three steps. First a set of n uncorrelated, unit length random degrees of freedom are scaled relative to each other by $b = \mathbf{\Lambda}a$. This can be thought of as generating a series of ellipses with principal axes aligned with the canonical basis vectors. Then each b represents a set of uncorrelated samples from Gaussian distributions such that b_i has standard deviation $\sqrt{\lambda_i}$. In the case that the matrix is positive definite, $\sqrt{\lambda_i}$ is always strictly greater than zero. The implication of this is that each random degree of freedom, in this representation, is linearly independent.

Multiplication by orthogonal matrices rotates a set of vectors while preserving the lengths and angles between the vectors [148]. The second step in generating a random field sample by this method is then multiplication of b by $\mathbf{Q}$ to rotate the scaled, uncorrelated vectors b to the correct correlation required for $Z(x, y)$. The rotation of the principal axes of the ellipses mentioned previously, relative to the canonical basis vectors, describes the correlation between the random variables of $Z(x, y)$. The shape of the ellipse also changes, becoming more eccentric as the rotation increases [249]. The third and final step is the addition of the scalar valued mean function, μ, to each point within the random field.

3.3.1.3 Random field simulation by Cholesky decomposition for positive definite covariance matrices

The Cholesky decomposition can be computed more quickly than an eigenvalue decomposition and thus can be used to improve the run time of a covariance matrix decomposition based random field generator. This is discussed in more detail in Section 3.4.

Given a symmetric, positive definite matrix $\mathbf{C}$, the Cholesky decomposition algorithm can be used to decompose $\mathbf{C}$ into a lower triangular matrix $\mathbf{L}$ such that:

$$\mathbf{C} = \mathbf{L}\mathbf{L}^{\mathbf{T}} \tag{3.48}$$

where T denotes the matrix transpose. Various algorithms implementing the Cholesky decomposition can be found in [148].

A random field can be generated using $\mathbf{L}$. After calculating $\mathbf{L}$ given equation (3.48), the random field $Z(x, y)$ can be obtained by:

$$Z(x, y) = \mathbf{L}a + \mu \tag{3.49}$$

Where a is a vector of size n whose entries are given by random samples of a standard normal distribution. Equation (3.49) gives a single value of a random field per finite element. To generate additional random field realisations for Monte Carlo Simulation it is necessary to retain $\mathbf{L}$ between simulations. For each realisation, a new vector of random samples a is generated and multiplied with $\mathbf{L}$ as per equation (3.49).

In a rough sense, the Cholesky decomposition takes the 'square root' of a matrix (although the square root of a matrix is not necessarily equal to $\mathbf{L}$ [148]). Recall from the previous Section that the eigenvectors of $\mathbf{C}$ can be thought of as representing the principal axes, with magnitude σ, of a set of ellipses. In contrast, $\mathbf{L}$ describes the same ellipses in a different form. The column and row vectors of $\mathbf{L}$ are the direction of the linear regression lines of one column vector over all others [326, 249, 129]. Let $Z(i)$ denote the value of the random field associated with the element in the discretisation mesh with index i. Also, let L_i denote the $i-th$ row vector of $\mathbf{L}$. Then, more simply, each L_i describes the random variable associated with $Z(i)$. If the correlation between some $Z(i)$ and $Z(j)$ is ρ, then correlation between these degrees of

freedom is equal to:

$$\rho_{x,y} = \cos\theta \tag{3.50}$$
$$\phi = \frac{\pi}{4} + \frac{\theta}{2}$$

where θ is the angle between the two possible regression lines describing the ellipses for $Z(i)$ and $Z(j)$ and ϕ is the angle between vectors L_i and L_j [73, 129].

To see how a random field is constructed by equation (3.49), consider the following. $\mathbf{L}$ is a lower triangular matrix and so for a row j, all entries in columns greater than j are zero. Consider the result of $Z(x,y) = \mathbf{L}a$. The first two entries will be:

$$Z(x,y)_1 = \mathbf{L}_{1,1} \times a_1 + 0$$
$$Z(x,y)_2 = \mathbf{L}_{2,1} \times a_1 + \mathbf{L}_{2,2} \times a_2 + 0$$

For a finite element-type basis functions, the Cholesky Decomposition method for random field simulation has a clear interpretation. As the ordering of mesh elements is arbitrary, the row ordering of $\mathbf{C}$ should not impact on the final result. The first degree of freedom in the random field is taken up by a_1, the choice of which element is represented by a_1 is arbitrary. The second degree of freedom also represents an arbitrary element. Ignoring other elements in the random field, a_2 only needs to be correlated to itself and the first degree of freedom. Proceeding in this manner, the uncorrelated a values are assigned the correct correlation structure. In practice, the row and column structure of matrices may impact numerical performance [148].

3.3.1.4 Random field simulation by Cholesky decomposition for positive semi-definite (singular) covariance matrices

The eigenvalues of the covariance matrix, $\mathbf{C}$ may become zero, or even negative, as a result of finite precision arithmetic and associated rounding errors. Additionally, even in theoretical infinite precision mathematics, if the covariance function is constant then linear dependencies between the columns of $\mathbf{C}$ may occur and $\mathbf{C}$ will be positive semi-definite. As a result, the covariance

matrix may become singular, or will be in the case of linear dependence. In this Section a method to generate random fields in the event that the covariance matrix is singular is presented. An analysis of the errors of this method is presented in Section 3.4.2. Methods exist to ensure that the covariance matrix is positive semi-definite. Naïve modifications to the Cholesky decomposition matrix can remove negative eigenvalues by adding a small error term to the main diagonal. This ensures diagonal dominance. More sophisticated variations on such a technique and the associated error bounds are detailed in [112, 66].

With negative eigenvalues removed, the $n \times n$ covariance matrix, $\mathbf{C}$, may be positive semi-definite and therefore singular. A method for resolving this problem is given in [17, 158], which is discussed here. A modification of the results in [17] to allow generating random field samples when the covariance matrix is singular is presented. First, recall that by equation (3.43), the number of linearly independent columns of $\mathbf{C}$ is equal to $n-k$ where k is the multiplicity of zero eigenvalues, as in equation (3.40). To find a useful decomposition of $\mathbf{C}$, it is possible to split the singular and non-singular parts of $\mathbf{C}$. The following procedure is from [17]. First reorder the columns of $\mathbf{C}$ such that the first $p = n-k$ columns are linearly independent. Then it is possible to write:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_I & \mathbf{C}_D \end{bmatrix}$$

where $\mathbf{C}_I$ is an $n \times p$ matrix and $\mathbf{C}_D$ is $n \times (n-p) = n \times k$ so:

$$\mathbf{C}_D = \mathbf{C}_I \mathbf{B}$$

where $\mathbf{B}$ is size $p \times k$. Also, $\mathbf{C}_I$ can be split into an upper $p \times p$ part, $\mathbf{C}_{IU}$, and a lower $k \times p$ part, $\mathbf{C}_{IL}$. Then:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{IU} & \mathbf{C}_{IU}\mathbf{B} \\ \mathbf{C}_{IL} & \mathbf{C}_{IL}\mathbf{B} \end{bmatrix}$$

Then, as $\mathbf{C}$ is symmetric, $\mathbf{C}_{IL}^T = \mathbf{C}_{IU}\mathbf{B}$ and then finally:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{IU} & \mathbf{C}_{IU}\mathbf{B} \\ \mathbf{B}^T\mathbf{C}_{IU}^T & \mathbf{B}^T\mathbf{C}_{IU}^T\mathbf{B} \end{bmatrix} \tag{3.51}$$

In this form, $\mathbf{C}$ can be made from just two parts. $\mathbf{C}_{IU}$ is a non-singular $p \times p$ matrix and $\mathbf{B}$ contains the coefficients describing the linear dependencies between the columns of $\mathbf{C}_{IU}$. Numerical examples are given in [17, 158]. To find $\mathbf{C}_{IU}$ and $\mathbf{B}$, the first step is to find the linearly dependent columns of $\mathbf{C}$. From Lemma 12.2 of [155], two column vectors, x and y are linearly dependent if:

$$\begin{vmatrix} x \cdot x & x \cdot y \\ y \cdot x & y \cdot y \end{vmatrix} = 0 \tag{3.52}$$

where $|\mathbf{A}|$ is the determinant of $\mathbf{A}$ and $x \cdot y$ is the scalar product of x and y.

The linear dependencies in $\mathbf{C}$ can be found by comparing columns from left to right. Let the column vectors of $\mathbf{C}$ be $C_1, \ldots C_n$. Start by comparing C_1 with all columns C_i for $1 < i \leq n$. Then compare C_2 with all C_i for $2 < i \leq n$. Once a C_i has been found to be linearly dependent, it can be excluded from subsequent checks. Continuing in this fashion, the linear dependence of all columns can be checked. If two columns, say C_j and C_k with $j < k$ are found to be dependent, a new row is made in $\mathbf{B}$ to represent column C_j and a new column to represent C_k. The corresponding entry into $\mathbf{B}$ is the factor ω_{jk}, from:

$$C_k = \omega_{jk} C_j \tag{3.53}$$

After the entries of $\mathbf{B}$ are filled in, $\mathbf{C}_{IU}$ is formed by deleting k dependent rows and columns from $\mathbf{C}$. Specifically, for each column C_k that is linearly dependent on a C_j with $j < k$, delete both row and column k from $\mathbf{C}$.

A random field sample can now be generated using $\mathbf{C}_{IU}$ and $\mathbf{B}$. For the non-singular part $\mathbf{C}_{IU}$, the procedure in Section 3.3.1.3 can be applied. Let $\mathbf{L}_{IU}$ be the lower triangular matrix produced by applying the Cholesky decomposition to $\mathbf{C}_{IU}$. Then, as in equation (3.49), $Z(x, y) = \mathbf{L}_{IU} a_I + \mu$ where a_I is of size $p = n - k$. This gives part of the random field sample.

To complete the sample, the linear dependencies described by $\mathbf{B}$ must be included. From the discussion in Section 3.3.1.3, the degree of correlation between two random degrees of freedom is represented in the covariance matrix by the angle between the two representative basis vectors. When the basis

vectors are linearly dependent they are, by definition, co-linear. Then linear dependence between basis vectors also indicates perfect correlation between these degrees of freedom. As in equation (3.53), let each C_k be linearly dependent on some C_j. Then $\mathbf{B}^T\mathbf{L}_{IU}a_I + \mu$ gives the value of the random field for each of the k linearly dependant elements.

Let $Z(i)$ be the value of the random field over the discrete element with index i. Then a random field can be generated from a singular covariance matrix by a modified Cholesky decomposition by:

$$Z(i) = \mathbf{L}_{IU}a_I + \mu \tag{3.54}$$

$$Z(k) = \mathbf{B}^T\mathbf{L}_{IU}a_I + \mu \tag{3.55}$$

where the $Z(i)$ entries come from the non-singular degrees of freedom contained in $\mathbf{C}_{IU}$ and the $Z(k)$ are linearly dependant on the $Z(i)$ parts of the random field sample.

In practice, it is preferable to first run a rank-revealing Cholesky decomposition as in [148, 178]. Such a decomposition fills in the entries of $\mathbf{L}_{IU}$, reordering the columns $\mathbf{C}$ as necessary, until the diagonals of $\mathbf{L}_{IU}$ become sufficiently small. Once this cut off, ϵ, is reached after $\text{rank}(\mathbf{L}_{IU}) \approx p$, the remaining k columns can be scanned for linear dependencies to fill in the entries of the $\mathbf{B}$ matrix. If $\mathbf{B}$ is filled in first, then there may be no linear dependencies and therefore computer time is wasted for no gain. By computing a rank-revealing Cholesky decomposition first, if $\mathbf{C}$ is non-singular so $\text{rank}(\mathbf{L}_{IU}) = n$, then there is no need to expend computation effort searching for linear dependencies.

3.3.1.5 Gaussian random field simulation examples

To illustrate the random field theory discussed so far, this Section presents a number of simulations of Gaussian random fields. All random fields presented were simulated using the Cholesky decomposition technique described in Section 3.3.1.3.

Figure 3.2 demonstrates the effect of isotropic correlation structures on Gaussian random field samples with mean zero, point marginal variance $\sigma^2 = 1$ and correlation function $C(s,t)$. Figures 3.2a and 3.2b demonstrate the effect of

correlation length scaling for the correlation function based on the exponential decay of the (standard Euclidean) $\mathcal{L}^2$ distance between points $s = (s_x, s_y)$ and $t = (t_x, t_y)$:

$$C(s,t) = \exp\left(\frac{-\|s-t\|_2^2}{\theta^2}\right) \tag{3.56}$$

for correlation length θ. For correlation lengths $\theta = 1$ and $\theta = 2$ for Figures 3.2a and 3.2b respectively, the random field samples are seen to become more smooth with higher correlation length. Figure 3.2c is a random field sample with the correlation function:

$$C(s,t) = \exp\left(\frac{-\|s-t\|_1}{\theta^2}\right) \tag{3.57}$$

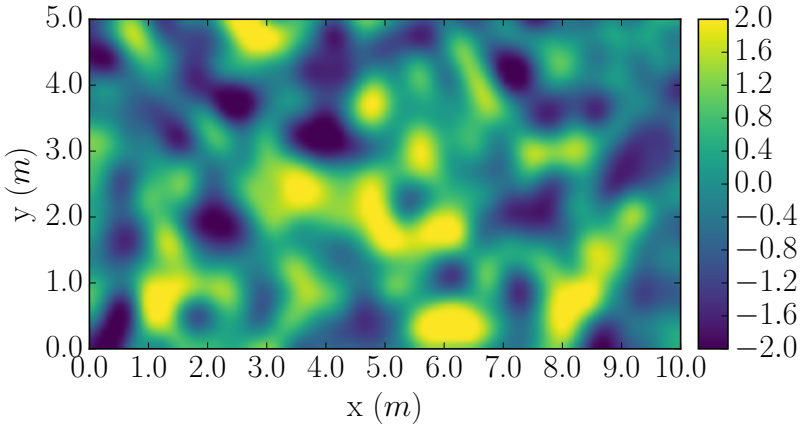
with $\theta = 2.0$. This is an exponentially decaying correlation based on the $\mathcal{L}^1$ distance between the points in the random field domain. In comparison to Figure 3.2b, the random field sample in Figure 3.2c shows high correlations in the directions aligned with the x and y axes, as would be anticipated when the $\mathcal{L}^1$ distance between points is considered.

Figure 3.3 demonstrated the effect of anisotropy of the correlation function on random field simulation. Each random field in Figure 3.3 is generated using the correlation function between points $s = (s_x, s_y)$ and $t = (t_x, t_y)$:

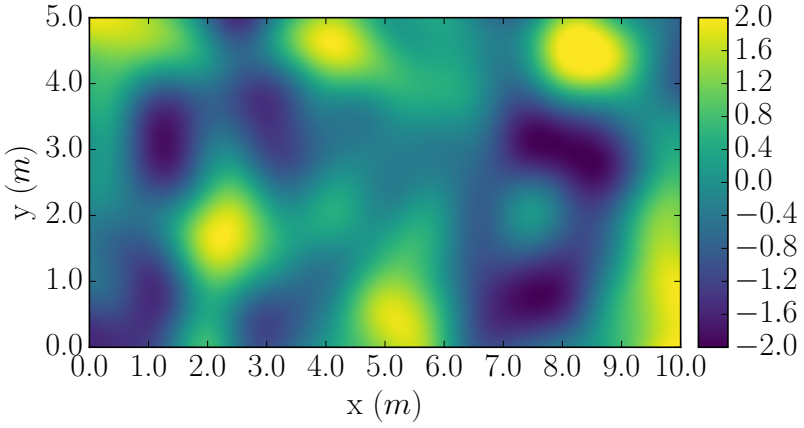
$$C(s,t) = \exp\left(\frac{-\|s_x - t_x\|_2^2}{\theta_x^2} + \frac{-\|s_y - t_y\|_2^2}{\theta_y^2}\right) \tag{3.58}$$

where θ_x and θ_y are termed the x and y correlation length parameters.

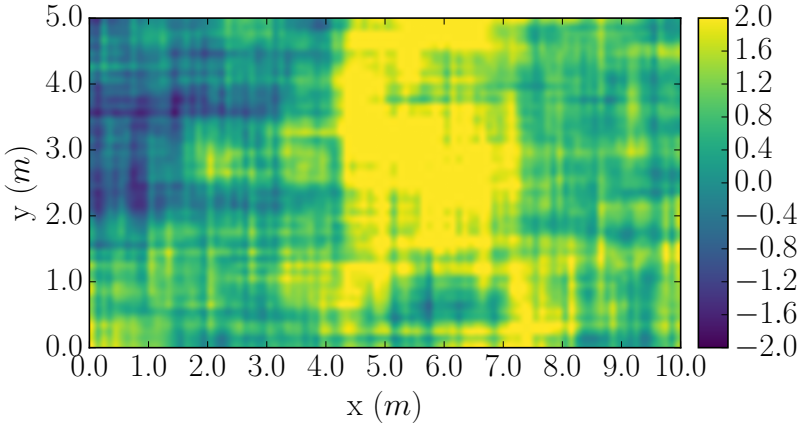
Figure 3.3a demonstrates a random field sample with high horizontal correlation described by $\theta_x = 10.0$ and $\theta_y = 1.0$. Figure 3.3a demonstrates a random field sample with high vertical correlation using $\theta_x = 1.0$ and $\theta_y = 2.5$. Spatial similarity structures, such as those found in soil masses, can be modelled by introducing anisotropy in the correlation function [118]. Figure 3.3c demonstrates a random field with a rotated anisotropic structure. This random field was simulated by calculating the correlation structure using points rotated 30 degrees from (x, y) to (x^r, y^r) and taking the correlation lengths aligned to the rotated principal axis as $\theta_x^r = 5.0$ and $\theta_y^r = 1.0$. By introducing rotations in the random field structure, it is possible to model a more broad range of phenomena than if the anisotropy is limited to particular axes.



(a) $C(s,t) = \exp\left(\frac{-\|s-t\|_2^2}{\theta^2}\right)$ for $\theta = 1.0$

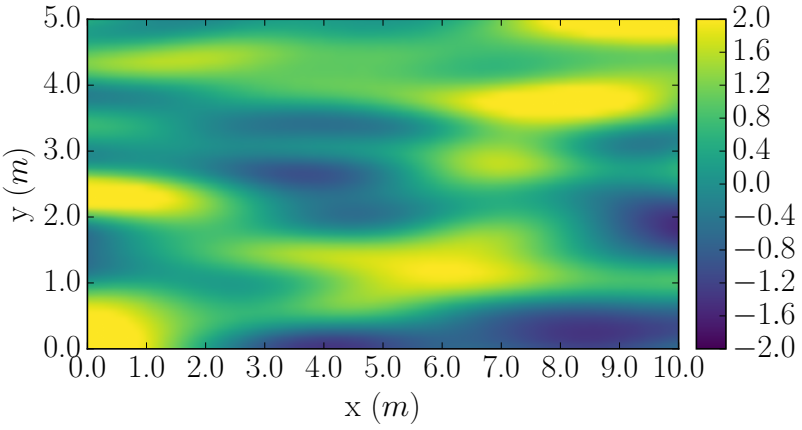


(b) $C(s,t) = \exp\left(\frac{-\|s-t\|_2^2}{\theta^2}\right)$ for $\theta = 2.0$

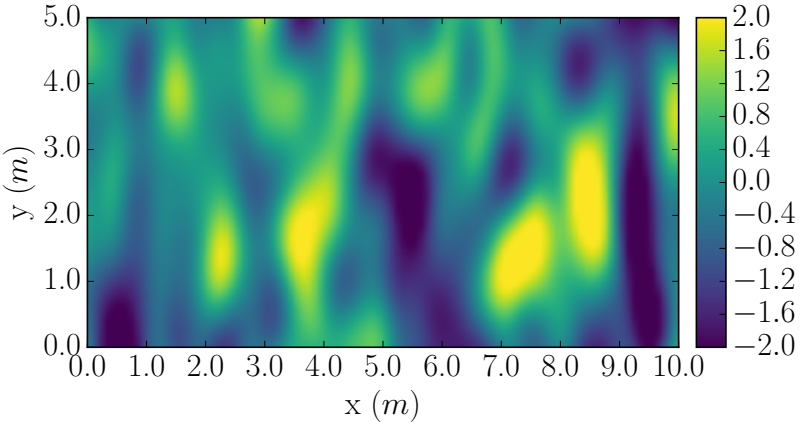


(c) $C(s,t) = \exp\left(\frac{-\|s-t\|_1}{\theta^2}\right)$ for $\theta = 2.0$

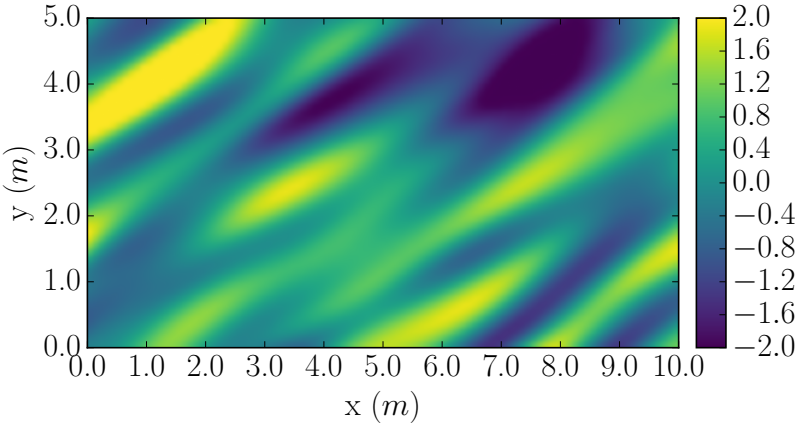
Figure 3.2: Gaussian random field simulation examples showing the effect of correlation length scaling. Figures 3.2a and 3.2b use an exponentially decaying $\mathcal{L}^2$ distance autocorrelation function with correlation lengths $\theta = 1$ and $\theta = 2$ respectively. Figure 3.2c demonstrates an exponentially decaying $\mathcal{L}^1$ distance correlation function.



(a) $C(s,t) = \exp\left(\frac{-\|s_x - t_x\|_2^2}{\theta_x^2} + \frac{-\|s_y - t_y\|_2^2}{\theta_y^2}\right)$ for $(\theta_x, \theta_y) = (10.0, 1.0)$



(b) $C(s,t) = \exp\left(\frac{-\|s_x - t_x\|_2^2}{\theta_x^2} + \frac{-\|s_y - t_y\|_2^2}{\theta_y^2}\right)$ for $(\theta_x, \theta_y) = (1.0, 2.5)$



(c) $C(s,t) = \exp\left(\frac{-\|s_x^r - t_x^r\|_2^2}{\theta_x^2} + \frac{-\|s_y^r - t_y^r\|_2^2}{\theta_y^2}\right)$ for $(\theta_x^r, \theta_y^r) = (5.0, 1.0)$

Figure 3.3: Gaussian random field simulation examples showing the effect of correlation length anisotropy. Figures 3.3a and 3.3b respectively demonstrate increased horizontal and increased vertical correlation lengths. Figure 3.3c demonstrates anisotropic correlation on a rotated axis.

3.3.2 Non-Gaussian Random Field Simulation

Non-Gaussian random fields are often required for modelling the types of parameters encountered in practice. For example, in Civil Engineering, material strengths are often strictly non-negative parameters. Examples of such parameters include Young's Modulus and yield stresses [265]. This Section demonstrates examples from two non-Gaussian random field methods useful for simulating continuous random fields based on transformations of underlying Gaussian fields. Specifically, simulations of lognormal random fields and arbitrary continuous point marginal distribution fields via copula theory are presented. Note that when using copula theory, the pointwise marginal distribution is not required to be non-negative, only continuous (as discussed in Section 3.2.1.3).

Lognormal random fields are frequently used in spatial statistical modelling to generate non-negative parameters and are detailed in [118, 237]. Lognormal random variables are random variables, X, such that their logarithm is normally distributed:

$$\ln(X) \sim \mathcal{N}\left(\mu, \sigma^2\right) \tag{3.59}$$

Given an underlying sample from a Gaussian random field, β, samples from a lognormal random field can be generated by taking the entrywise exponential of the vector of values representing the Gaussian random field sample (see [167]):

$$\beta \sim \mathcal{N}(\mu, \Sigma) \tag{3.60}$$

$$\gamma \sim \exp\left(\beta\right) \tag{3.61}$$

such that $\gamma_i = \exp\left(\beta_i\right)$.

Copula theory, discussed in Section 3.2.1.3, can be used to generate samples from a non-Gaussian random field. Of particular interest in this thesis are those fields with a Gaussian correlation structure (that is, a Gaussian copula) but non-Gaussian pointwise marginal distributions. When considering spatially distributed data, pointwise marginal measurements can be combined with the Gaussian copula spatial similarity model given by equation (3.36). Simulation of non-Gaussian random fields via copula theory is described in this Section. First, a Gaussian sample is drawn from a correlated Gaussian distri-

bution. Applying the inverse normal transform to the marginals of the Gaussian sample recovers the desired copula. Finally, the desired marginals can be used to construct the final non-Gaussian random field sample by applying the inverse of the marginals to the copula sample as in equation (3.34).

Mathematically, the process can be described by transforming a sample from Gaussian random field, β to a sample from the desired non-Gaussian field, η, with the desired marginal cumulative distribution F. The copula simulation method proceeds by first sampling the vector β from a zero mean Gaussian random field with the appropriate correlation structure:

$$\beta \sim \mathcal{N}(0, \Sigma) \tag{3.62}$$

Then, to recover the copula distribution, the cumulative Gaussian Φ (also related to the error function, see §2 [224] for details) is applied entrywise to β:

$$\xi = \Phi(\beta) \tag{3.63}$$

such that $\xi_i = \Phi(\beta_i)$. Finally, the desired marginal distribution, F, is recovered by mapping from the copula sample, ξ, to the final distribution sample by applying the inverse cumulative distribution, F^{-1}, to the entries of ξ:

$$\eta = F^{-1}(\xi) \tag{3.64}$$

such that $\eta_i = F^{-1}(\xi_i)$. As the entries of ξ are from the copula distribution, they are samples from uniform distributions. Then $F^{-1}(\xi_i)$ is a sample with the distribution described by F as this is a form of inverse transform sampling (see §2.7.2 of [224]), as per the theoretical developments in Section 3.2.1.3.

3.3.2.1 Numerical Demonstrations

Non-Gaussian random field simulations are demonstrated in this Section. Figure 3.4 demonstrates a number of distributions used as the pointwise marginal distributions for the random field simulations in Figure 3.5. The Gaussian distribution is included in Figure 3.4 for comparison. The distributions in Figure

3.4 are:

$$P(x) = \mathcal{N}(\mu, \sigma^2) \qquad \text{Normal}(\mu, \sigma^2) \tag{3.65}$$

$$P(x) = \exp(\mu + \sigma\mathcal{N}(0,1)) \qquad \text{Lognormal}(\mu, \sigma^2) \tag{3.66}$$

$$P(x) = \frac{\lambda^\alpha x^{\alpha-1} e^{-\lambda x}}{\Gamma(\alpha)} \qquad \text{Gamma}(\alpha, \lambda) \tag{3.67}$$

$$P(x) = 2\mathcal{N}(0,1)\Phi(\alpha x) \qquad \text{Skew-Normal}(\alpha) \tag{3.68}$$

where $\Gamma(\alpha)$ is the Gamma function (see equation (2.20) of [224]) and $\Phi(x)$ is the cumulative normal distribution (see §A of [323]). For further details of the Gamma distribution see §2.6.4 of [224]. For details regarding the skew-normal distribution, see [286, 176, 18]. Note that the Gamma and lognormal distributions are non-negative.

Figure 3.5 demonstrates a number of non-Gaussian random fields. Each random field was generated over a $10m \times 5m$ region using an underlying Gaussian distribution mean zero, pointwise variance $\sigma^2 = 1$ and Gaussian copula correlation structure derived from the correlation function:

$$C(s,t) = \exp\left(\frac{-\|s-t\|_2^2}{\theta^2}\right) \tag{3.69}$$

for $\theta = 1.0$.

Figure 3.5a shows a lognormal random field generated as per equations (3.60) and (3.61). Figure 3.5b demonstrates a Gaussian copula random field with Gamma distribution point marginals for parameters $\alpha = 1$, $\lambda = 1$. Figure 3.5c demonstrates a Gaussian copula random field with Skew-Normal distribution point marginals for parameters $\alpha = -1$. The non-Gaussian fields in Figures 3.5b and 3.5c were generated as per the method described in equations (3.62), (3.63), (3.64).

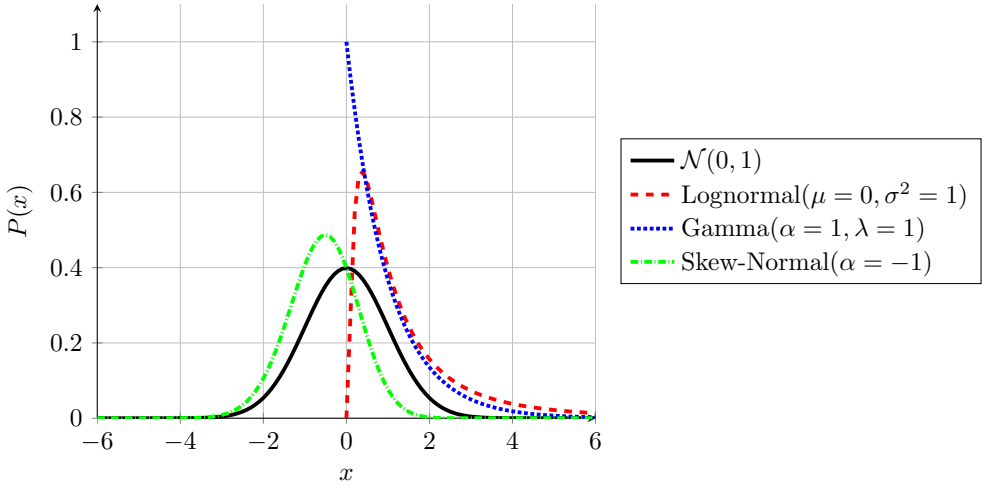
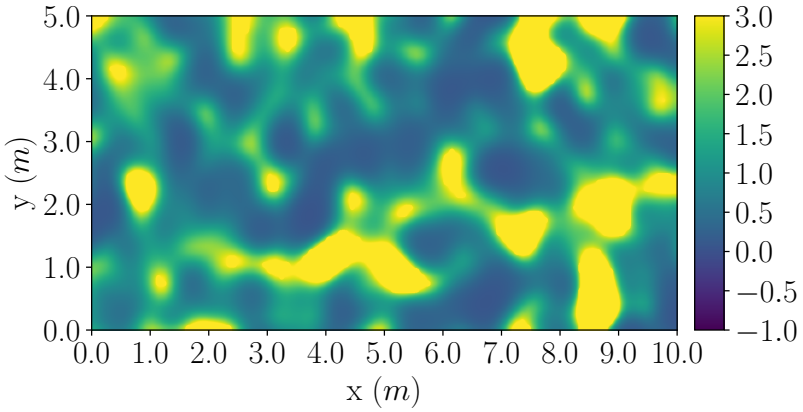
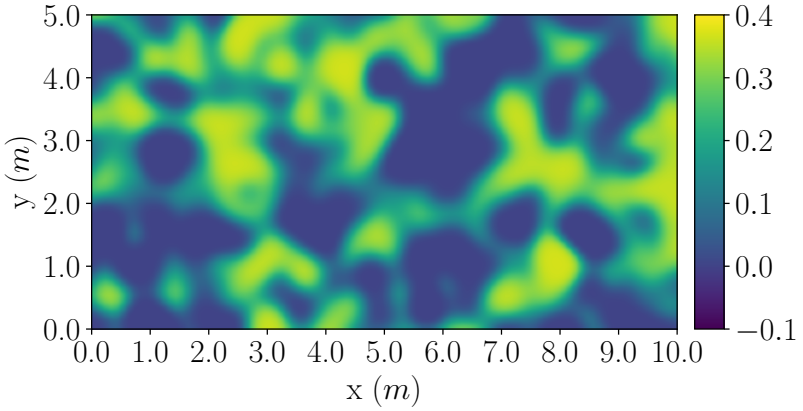


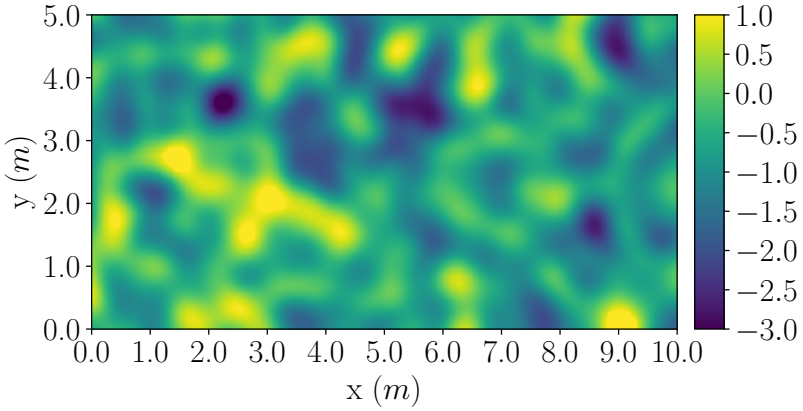
Figure 3.4: Example point marginal probability distributions for non-Gaussian random field simulations in Figure 3.5 and, for comparison, the Gaussian distribution. The distributions shown are the lognormal distribution (used for Figure 3.5a), the gamma distribution (used for Figure 3.5b) and the skew-normal distribution (used for Figure 3.5c). The formulas for these distributions are shown as equations (3.65), (3.66), (3.67), (3.68).



(a) Lognormal random field simulation with point marginal: $\exp(\mathcal{N}(0,1))$



(b) Gamma random field simulation with point marginal: Gamma($\alpha = 1, \lambda = 1$)



(c) Skew-Normal random field simulation with point marginal: Skew-Normal($\alpha = -1$)

Figure 3.5: Non-Gaussian random field simulation examples using Gaussian copula correlations. The type of random field refers to the pointwise marginal distribution. All simulations generated by transformations of a Gaussian random field $\mathcal{N}(0, \Sigma)$ (see Section 3.3.2) with the correlation function in equation (3.69). Figures 3.5a, 3.5b and 3.5c are sampled respectively from lognormal ($\exp[\mathcal{N}(0, \Sigma))]$, equation (3.66)), Gamma($\alpha = 1, \lambda = 1$) (equation (3.67)) and Skew-Normal($\alpha = -1$) (equation (3.68)). Note the different ranges on the colour bars for each plot.

3.3.3 Phase field techniques for simulating arbitrary field features

One advantage of sampling based methodologies for Uncertainty Quantification is that random fields with complicated analytical expressions can be used for analyses as long as samples can be generated from the distribution of interest. This Section presents a phase-field threshold based technique for including non-smooth features in probabilistic input distributions by combining Poisson Point Processes and Bernoulli Processes with the smooth random field simulation methods presented earlier in this Chapter. Related work in this area is presented in [229].

Phase field models can be used to simulate combinations of material states by modelling the presence or absence of different states (or phases) by reference to an *order parameter*. Prominent applications of phase field models include multiphase fluid flows [53] and fracture modelling [207, 165, 348]. In the fluid flow case, the order parameter can be used to model the relative volumetric fraction of different fluids, defining the fluid boundary at a particular value of the order parameter. For example, in a mixed air and water simulation the order parameter, α, could be taken to range from $\alpha = 0$ (all air) to $\alpha = 1$ (all water) with the interface boundary defined as points with $\alpha = 0.5$. The multiphase dynamics can be simulated by modelling transport of the order parameter itself [53]. In the fracture mechanics case, the order parameter can be taken to represent fracture intensity, defining a distinction between intact and broken as well as crack tip locations.

Section 3.3.3.1 demonstrates a phase field modification to smooth random field simulation, representing a problem similar to, say, modelling aggregrate particles within a binder matrix. Circular inclusions over a smooth random field are simulated via a Poisson point process. In Section 3.3.3.2, a simulation of jointed rock is also presented. The jointed rock is modelled by a spatially autocorrelated lognormal random field of the intact rock strength. This model is combined with a Bernoulli Process to indicate the presence or absence of a rock defect intersecting a trace line in a particular location, emulating the type of information gleaned from core logging [187].

3.3.3.1 Modelling field inclusions by a Poisson Point Process

This Section demonstrates a phase field model of inclusions to a random field. In the particular example demonstrated, the two fields modelled (the base random field and the inclusions) are taken to represent the values of two separate scalar fields, $f_1(x)$ and $f_2(x)$ respectively. These presence or absence of these inclusions is modelled using the order parameter, $\alpha(x)$, of a phase field. The final estimated value of the sampled field is calculated using the volume fraction estimated by the order parameter such that:

$$f(x) = (1 - \alpha(x))f_1(x) + \alpha(x)f_2(x) \tag{3.70}$$

An example of an application of such a model in practice is the Volume of Fluid method for multiphase fluid flow [283, 162].

The *Poisson distribution* can be used to model the number of instances of some event occurring in a given interval (of say space or time). From §2 of [224], a random variable with a discrete probability mass function describing the number of times an event occurs, k, has a Poisson distribution if:

$$P(k) = e^{-\lambda}\frac{\lambda^k}{k!} \quad k \in \mathbb{N} \tag{3.71}$$

where the parameter λ is the average number of times an event occurs per interval.

From §4.6 of [90] and [261, 316], taking the events of interest to be situated at locations within some spatial domain, Ω, with volume, V, the Poisson point process can be used to model the probability of k point events within Ω as:

$$P(k) = \frac{(\lambda V)^k e^{-\lambda V}}{k!} \tag{3.72}$$

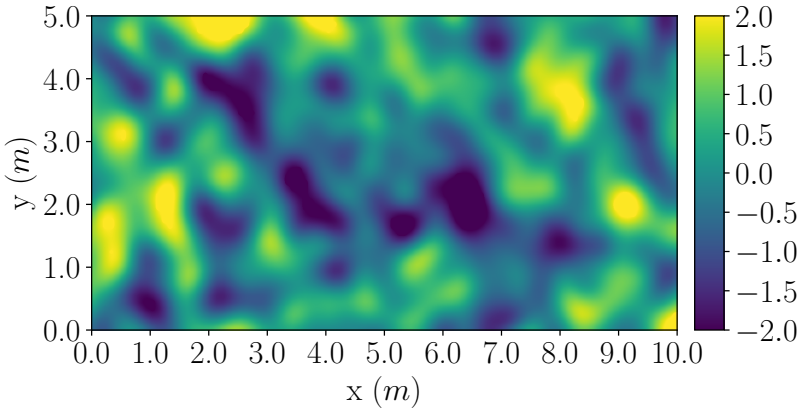
As such, a Poisson point process can be used to model inclusions to a field. Given the parameters for a Poisson distribution and the shape of the inclusions to be added, the Poisson distribution can be sampled from by the procedure discussed in [214] to generate a number, k, of inclusions. Then, the spatial domain can be sampled from uniformly k times. Each inclusion is placed at the k sampled locations by adding to the value of $\alpha(x)$ in the region defined by the inclusion. The phase field is then truncated to be between the values

of 0 and 1. Finally, the field sample $f(x)$ can be calculated as per equation (3.70).

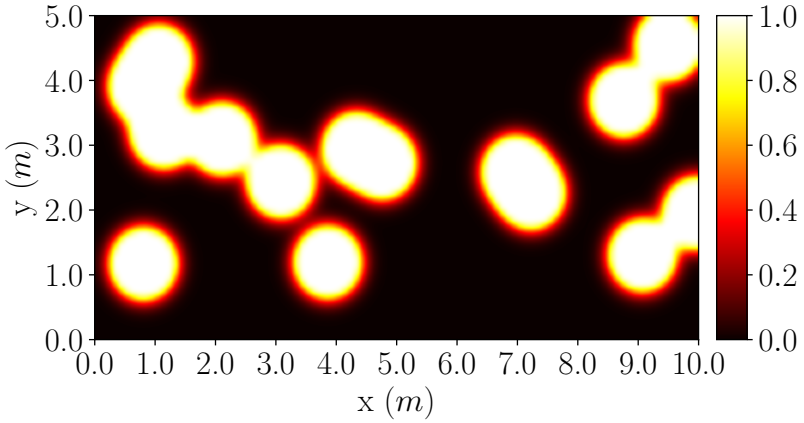
Figure 3.6 demonstrates a phase field model combining Gaussian random field (Figure 3.6a) with circular inclusions via an order parameter (Figure 3.6b) on a $10m \times 5m$ rectangular region, Ω discretised by a regular 200 by 100 grid. The combined field simulates the values of the function $f(x)$ which describes the pointwise scalar field fluctuations (relative to a mean value) over a spatial region (Figure 3.6c). A base Gaussian random field models the overall scalar fluctuations, $f_1(x)$. The random field is simulated as a having zero mean and correlation function:

$$C(s,t) = \exp\left(\frac{-\|s-t\|_2^2}{\theta^2}\right) \tag{3.73}$$

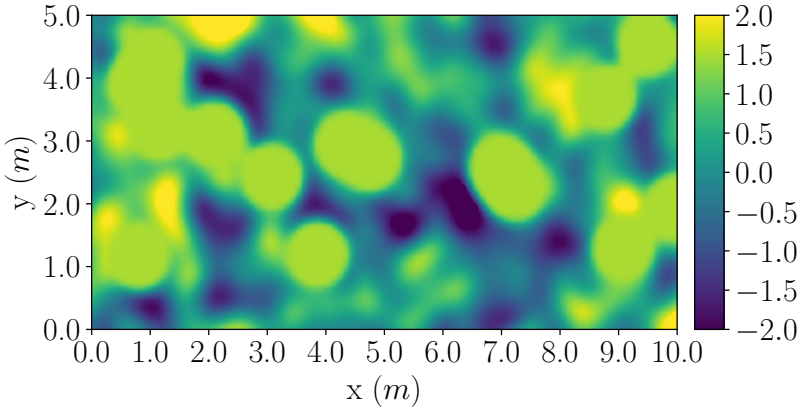
for $\theta = 1$. The presence or absence of inclusions are modelled using an order parameter field, $\alpha(x)$. The inclusion centres are modelled as a Poisson point process with $\lambda = 10$. Given a sample k from a Poisson distribution, k points, $(s,t) \in \Omega$, are sampled uniformly over Ω. The samples are used to represent the centre, (c_x, c_y), of the inclusions. The circular inclusions are represented by truncated, upscaled Gaussian density equal to $10.0 \times \mathcal{N}\left((c_x, c_y), r^2\right)$ where r is proportional to the radius of the inclusions. The circular inclusions are truncated by restricting $\alpha(x)$ to be between zero and one. The value of the field $f_2(x)$ is taken be a constant, $C = 1.5$. The scalar field $f(x)$ could represent, for example, material density in a matrix-with-aggregate combination model.



(a) Base Gaussian random field sample, $f_1(x)$



(b) Order parameter field $\alpha(x)$ after sampling $k = 15$ circular inclusions



(c) Estimated field $f(x) = (1 - \alpha(x))f_1(x) + \alpha(x)f_2(x)$ for $f_2(x) = 1.5$

Figure 3.6: Phase field scalar field simulation of combined random field and inclusion features model. Figure 3.6a demonstrates a sample from $f_1(x)$, a Gaussian random field with mean zero and correlation function from equation (3.73). Figure 3.6b demonstrates a sample of the order parameter field, $\alpha(x)$, after sampling $k = 15$ circular inclusions (modelled by scaled, truncated Gaussians). Figure 3.6c presents the final scalar field $f(x) = (1 - \alpha(x))f_1(x) + \alpha(x)f_2(x)$ for $f_2(x) = 1.5$.

3.3.3.2 Simple phase field jointed rock mass model

A phase field model of a sample of jointed rock mass can be generated in a similar manner as the model in Section 3.3.3.1. Consider a rock mass with several defect sets (see [187]), modelled as planes with uncertain orientations. Modelling of fractured rock is typically carried out using Discrete Fracture Networks (DFNs) [56, 263, 107]. These methods describe the geometric structure of the defect planes present. The model presented in this Section uses a phase field defect intensity model that is more suitable than DFNs for later inclusion in, for example, structural deformation analysis. Although such structural modelling is not presented, given a rock mass simulation technique, it is simple to incorporate new spatial material models into a sampling based Uncertainty Quantification analysis. DFN methods attempt to model the exact geometry of the defects, while the phase field model presented in this Section models only the relative intensity of defects within a volume. In this sense, there is a parallel between Eulerian and Lagrangian descriptions of flow fields in continuum mechanics [150, 250]. The Lagrangian, material tracking model is philosophically aligned with the DFN model. The Eulerian, fixed control volume approach is related more closely to the phase field approach to discrete field inclusions.

In this Section, an example two dimensional rock mass is simulated. The simple model presented in this Section is not sufficient to make detailed mechanistic evaluations of fracture network responses. However, the techniques presented suggest directions for future work. As the example simulations presented are intended to be illustrative, a two, rather than three, dimensional simulation is modelled. The intact rock strength is modelled as a lognormal, spatially autocorrelated random field. The defects are modelled as a phase field, $\alpha(x)$, laid over the intact rock model. In two dimensions, only the dip (or slope) relative to the viewing angle needs to be modelled. The defect dip for each defect set is estimated, in this example, by Gaussians. To simulate the defect intensity, consider a line passing through a rock mass (simulating the drill path from a core logging procedure [187]). Intersections of defect sets with this line can be considered to be a Bernoulli process (defined below). Given a set of intersections for each defect set, defect dips can be sampled from probabilistic models of the defect set dips. Given a sampled intersection location and dip angle, defects can be modelled as line inclusions in a phase

field, $\alpha(x)$. The line inclusions can be modelled using a Gaussian density on the perpendicular distance of the point, x, to the line. That is, for a point x, the $\alpha(x)$ value is proportional to:

$$\alpha(x) \propto \exp\left(-1\frac{(d_p)^2}{2\delta^2}\right) \tag{3.74}$$

where d_p is the perpendicular distance from x to the line. The exponentially decaying influence width of the line is controlled by the term δ. Given a random field, $f_1(x)$, of the material strength, the defect indicator phase field $\alpha(x) \in [0,1]$ and the defect strength modification field, $f_2(x)$, the rock strength at a location can be simulated by:

$$f(x) = (1-\alpha(x))f_1(x) + \alpha(x)f_2(x) \tag{3.75}$$

Although this is a simple model of spatially distributed rock mass strengths, it indicates a path to more complex phase field synthetic rock simulations.

Following §10.5 of [90], a Bernoulli process is defined as a sequence of random variables, $\{X_i\}_{i=1}^{\infty}$, such the value of the random variable X_i is 0 or 1 and the probability for X_i to be 1, $P(X_i = 1) = p$, is the same for all random variables in the sequence. Samples can be drawn from a Bernoulli process by taking a finite vector, β, and then assigning a value of 0 or 1 with probability p for each vector entry, β_i. Computationally, this can be done by sampling a value $u \sim U([0,1])$ from a random number generator and setting:

$$\beta_i = \begin{cases} 1 & \text{if } u < p \\ 0 & \text{otherwise} \end{cases} \tag{3.76}$$

For modelling defects, the intensity of defect intersections is reflected in the probability, p. Defects crossing the intersection line in the simulated rock mass are assumed to continue to the edge of the spatial domain modelled. Although the defect extent within the rock is unknown and not modelled accurately by this method, it is typically the case that it is very difficult to estimate and then subsequently accurately model the length and volume characteristics of defects. This is also the case for DFN models [243].

Figure 3.7 demonstrates a sample from a phase field simulated rock mass, on a $10m \times 5m$ rectangle, Ω, discretised by a regular 200 by 100 grid. The

intact rock strength is modelled as a lognormal random field, $f_1(x)$, taken as the exponential of an underlying Gaussian random field with mean zero and correlation matrix calculated using the correlation function:

$$C(s,t) = \exp\left(\frac{-\|s-t\|_2^2}{\theta^2}\right) \tag{3.77}$$

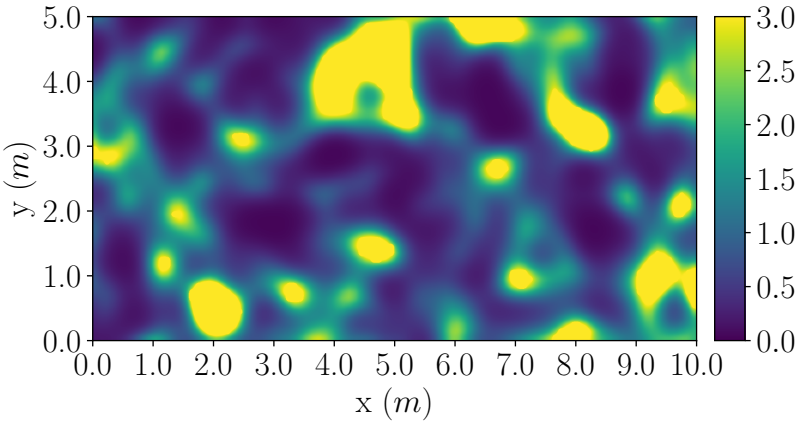
for $\theta = 1.0$. The base random field sample is shown as in Figure 3.7a. Two defect sets are modelled (in angular units of degrees) as:

- Defect set A apparent dip: $\mathcal{N}(-60, 10^2)$
- Defect set B apparent dip: $\mathcal{N}(5, 3^2)$

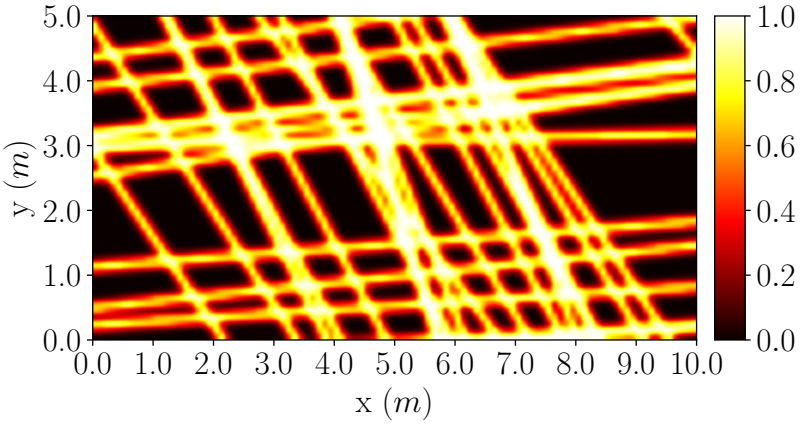
The intensity (to intersect the vertical line through the centre of the rectangle Ω for each defect set) is taken as:

- Defect set A intersection intensity: $p_A = 0.06$
- Defect set B intersection intensity: $p_B = 0.1$

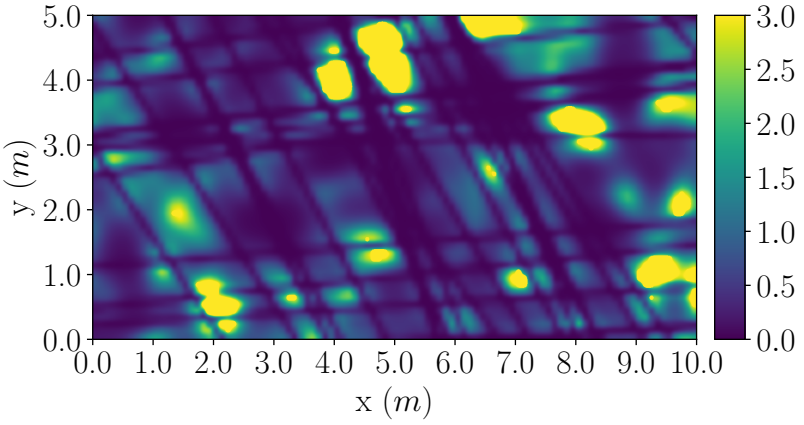
The defect phase field sample is shown as Figure 3.7b. A defect influence width of $\delta \approx 0.11$ (approximately 1.5 times the radius of the discrete grid squares) was used for the simulation. The defect strength field value was set to a constant $f_2(x) = C$ with $C = 0.01$. Figure 3.7c shows the final rock strength field calculated using equation (3.75). More advanced simulations of jointed rock masses could include physical fracture models to estimate crack propagation within rock (or other quasi-brittle material such as concrete). This would be a worthwhile topic for future research. In particular, to extend the phase field model to mechanistic analysis of jointed rock masses, it would be useful to include information about joint roughness, strength and stress states. Joint shear strength estimation models, specifically the Barton model (summarised in §4.2.6 of [143]) based on Joint Roughness Coefficient (JRC) and Joint wall Compressive Strength (JCS) could be utilised for such simulations in the future.



(a) Base Lognormal random field sample, $f_1(x)$



(b) Order parameter field $\alpha(x)$ after sampling defect set inclusions



(c) Estimated field $f(x) = (1 - \alpha(x))f_1(x) + \alpha(x)f_2(x)$ for $f_2(x) = 0.01$

Figure 3.7: Scalar Phase field field simulation of jointed rock mass random field. Figure 3.7a shows the intact rock strength as a lognormal random field $f_1(x)$ sampled from $\exp(\mathcal{N}(0, \Sigma))$ with correlation function from equation (3.77). Figure 3.7b is a sample of the order parameter field, $\alpha(x)$, generated from two defect sets with orientations $\mathcal{N}(-60, 10^2)$ and $\mathcal{N}(5, 3^2)$ and intersection intensities $p_A = 0.06$ and $p_B = 0.1$. Figure 3.7c presents the final rock strength field $f(x)$ for $f_2(x) = 0.01$.

3.3.4 Discussion

This Section presented a series of random field simulation techniques suitable for modelling spatially autocorrelated fields. Example simulations of each of these random field models were demonstrated. Gaussian random fields were simulated based on covariance matrix techniques. Techniques for simulating non-Gaussian random fields based on copula theory were discussed. Finally, phase field simulation of combination fields, suitable for empirical models of physical phenomena, were presented. All of these simulation techniques can be combined with Monte Carlo Simulation for Uncertainty Quantification for physical systems modelling.

3.4 Computational complexity of PDE Monte Carlo Analysis combined with covariance matrix decomposition

3.4.1 Computational complexity considerations for Uncertainty Quantification

For Uncertainty Quantification analysis, the combination of a PDE solver with random field models is computationally intensive. Applying Monte Carlo Simulation, for example, to the estimation of output space probability distributions and Quantities of Interest requires sampling from the input probability distribution, feeding the inputs into a PDE solver and using the output of this solver to generate samples from the output distribution. The Random Finite Element Method (RFEM) [118], for example, combines random field simulation with finite element analysis to perform Uncertainty Quantification. Such an approach relies on the generation of a very large number of both random field samples and the solution of finite element problems. Assessing the computational requirements and accuracy is a useful tool to check that a given procedure is feasible. This Section discusses the numerical accuracy and computational complexity of covariance matrix decomposition based random field sampling, following the authors published analysis in [158].

These results are discussed with reference to an analysis of the computational complexity of Stochastic PDE MCS with random field inputs based on co-

variance matrix decomposition simulation of a spatial field. In particular, as FEM is used extensively in this thesis, the computational complexity analysis is discussed using the Finite Element Method as a representative of a PDE solver for MCS. It is noted, however, that the solution of PDE problems can be framed as an optimisation problem [198]. Optimisation (and search) problems are subject to the so-called *No Free Lunch Theorems* (NFL Theorems) [398, 396, 397] which essentially state that (for the types of optimisation problems of interest) the computational performance of any solution, averaged over all problems in a class, is the same for all solution methods. That is, no algorithm performs better than all others over all problems. The relevant implication of the NFL Theorems here is that, in the absence of prior knowledge about an optimisation problem, no strategy can be expected to perform better than another and as such a general purpose optimisation algorithm essentially impossible. That is, given an optimisation strategy that performs well on one type of problem, this algorithm is unlikely to perform better on a different problem than another algorithm that has been specifically designed for the second problem [185].

For the Uncertainty Quantification analyses in this thesis, this leads to several important implications when considering the computational complexity of the PDE solver selected for MCS based analysis. This rough discussion here is used to justify the computational complexity analysis of covariance matrix decomposition for MCS Uncertainty Quantification by considering FEM rather than all PDE solvers. The first implication of the NFL Theorems to be considered here is that if, for a given problem, an FEM-based method has ideal performance (in the sense of computational complexity and approximation accuracy) then another PDE method may not perform well. Specifically, any other PDE solver method will only be as good as or worse than FEM on average. The second implication is the converse of the first and follows directly from the NFL Theorems. If, for a given problem some given PDE method works well (for example Finite Volume representation based solvers, see [386]), then FEM may not be well suited to solving the problem. Specifically, FEM will perform on average as well as or worse than the more suitable method. Then, analysing the average to worst case computational performance of an FEM based MCS method for Uncertainty Quantification is abstractly equivalent (via the NFL Theorems and the discussion above) to analysing any PDE solver for MCS Uncertainty Quantification. If a suitable PDE method performs well on some

class of problems, it will have similar performance to the average to worst case performance of FEM on problems for which FEM-based analysis is known to be an efficient PDE solver method for MCS Uncertainty Quantification. As such, the asymptotic performance of any PDE solver method will be similar to the analysis presented here for FEM assuming that a PDE solver reasonably well suited to the problem of interest is considered. This is a rough analysis, but is intended to demonstrate that the computational complexity analysis presented here is applicable to a wide range of problems despite the fact that the detailed analysis later in this Section is presented with specific reference to FEM in order to suit the numerical analyses presented in the later Chapters of this thesis.

Before discussing these analysis, the following brief note on some required notation is necessary. To estimate the computational complexity of algorithms, it is useful to adopt big-$\mathcal{O}$ notation as in [80]. To count the time to complete an algorithm, each floating point operation (flop) is summed. An algorithm may take, for example, $\mathcal{O}(n^5) \approx 7n^5 + 6n^2$ flops to complete where n is the number of inputs. In big-$\mathcal{O}$ notation, coefficients are dropped and only the dominating terms as n approaches infinity are retained.

3.4.1.1 Computational complexity of covariance matrix decomposition

Both the Cholesky and eigenvalue decompositions take roughly $\mathcal{O}(n^3)$ flops [148]. However, Cholesky decomposition can be completed in approximately $n^3/3$ flops versus $4n^3/3$ for a typical tridiagonal QR based eigenvalue decomposition [148]. While the asymptotic runtime of the two algorithms is the same, the speed of the Cholesky decomposition is beneficial.

To generate a set of random fields for, say, Monte Carlo analysis using covariance matrix decomposition, the decomposition need only be completed once. After the initial $\mathcal{O}(n^3)$ decomposition, a random field sample can be generated rapidly. For example, with Cholesky decomposition, as in equation (3.49), the vector a and multiplication with the lower triangular matrix $\mathbf{L}$ is required. Then a can be generated in $\mathcal{O}(n)$ flops. Multiplication of a vector with a lower triangular matrix is also fast, taking just $\mathcal{O}(n^2)$ flops [148].

If the covariance matrix is singular, then the procedure in Section 3.3.1.4 can

be applied to find and delete linear dependencies. Calculating a scalar product of two vectors takes $\mathcal{O}(n)$ flops [148]. Then computing equation (3.52) takes $4\mathcal{O}(n) + 5 \approx \mathcal{O}(n)$ flops. To find all linear dependencies, each column must be iterated over from left to right, taking $\mathcal{O}(n)$ work. Then from column C_i, columns C_j for $i < j \leq n$ must be iterated over, also taking $\mathcal{O}(n)$. Then the total complexity of the column elimination algorithm is $\mathcal{O}(n^3)$ and is therefore not asymptotically any worse than the decomposition itself. If a parallel and block matrix approach was taken for very large n, then care would need to be taken to minimise the overhead associated with moving data, such as reordering columns.

The total computational complexity of random field generation by covariance matrix decomposition is then:

$$\mathcal{O}(n^3) + m \cdot \mathcal{O}(\text{FEM}) \cdot \mathcal{O}(n^2) \tag{3.78}$$

where m is the number of MCS iterations used for RFEM and $\mathcal{O}(\text{FEM})$ is the complexity of each finite element analysis. These components are discussed in the following Sections.

3.4.1.2 Computational complexity of finite element analysis

The computational complexity of finite element analysis is discussed in [353] and summarised in [113]. Linear FEM for elliptical problems (for example linear elasticity) takes approximately $\mathcal{O}(nw^2)$ where n is the number of input elements and w is the stiffness matrix bandwidth. The bandwidth term reflects the fact that stiffness matrices are often sparse, that is, mostly filled with zeroes. If plastic (hysteresis induced nonlinearity), or some other form of nonlinear FEM is used, then a number of iteration steps, l must be carried out. Then:

$$\mathcal{O}(\text{FEM}) = \mathcal{O}(nw^2 l) \tag{3.79}$$

where l is low for a convergent problem and higher for an unstable problem. These steps l, for a typical nonlinear FEM method, represent iterations of Newton's method [111]. When modelling plasticity (hysteresis) it is often necessary to include additional corrective steps when iterating Newton's Method

via the so-called return-mapping (see [98, 340] and §17 of [78]). These additional computations introduce additional computational complexities.

3.4.1.3 Computational complexity of RFEM

Combining equations (3.78) and (3.79), the following total complexity of RFEM is obtained:

$$\mathcal{O}(\text{RFEM}) = \mathcal{O}(n^3) + m \cdot \mathcal{O}(nw^2l) \cdot \mathcal{O}(n^2)$$

This can be simplified to:

$$\mathcal{O}(\text{RFEM}) = \mathcal{O}(mn^3w^2l) \tag{3.80}$$

It can then be noted that the computational complexity of RFEM by covariance matrix decomposition is not limited by the decomposition itself. It is the MCS finite element iterations which cause a computational bottleneck. Consider, for example, an FFT based random field generator. In two dimensions, this takes $\mathcal{O}\left(n^2 \log n\right)$ work [118, 80]. This work must be carried out for each of the k MCS iterations. In this case, $\mathcal{O}(\text{RFEM}) = k \cdot \mathcal{O}(n^3w^2l \log n)$. LAS has a similar run time complexity to FFT, taking 1.5 to 2 times longer per realisation in two dimensions [119]. Thus LAS has a similar asymptotic run time to FFT. As such, the choice of random field generator is not the computationally limiting factor for RFEM. Rather, the Monte Carlo FEM iterations are the most significant computational cost, especially for large m.

3.4.2 Precision of RFEM with covariance matrix decomposition

3.4.2.1 Preliminaries

Finite precision arithmetic carries round off errors. The numerical precision of covariance matrix decomposition random field sampling for use with RFEM is addressed. Specifically, an estimate of the roundoff errors associated with the generation of stationary random fields by Cholesky decomposition for RFEM is given.

The error analysis presented requires the concept of backward and forward errors. First, consider a simple calculation $y = f(x)$. Let $\hat{y}$ denote the finite precision approximation to y. Then $\hat{y} = f(x + \Delta x)$ and Δx is termed the backward error. Let $x + \epsilon$ be some measured quantity, where ϵ is the error in x. If $\Delta x \ll \epsilon$ then it is difficult to reject the computed $\hat{y}$ on the basis of finite precision rounding errors. Similarly, the forward error is given by $y - \hat{y} = \Delta y$.

Modern computers use the IEEE standard 754 for binary floating point arithmetic. The 64 bit double precision type is used for many analyses in this thesis. From [178], the unit roundoff, u, for a double is defined to be:

$$u = 2^{-53} \approx 1.11 \times 10^{-16} \tag{3.81}$$

The vector two norm and the matrix two and Frobenius norms are required. From [148], the vector two norm is defined, for some vector x of length n, as:

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{(1/2)} \tag{3.82}$$

Also from [148], the matrix two and Frobenius norms are, for some $n \times n$ matrix $\mathbf{A}$, defined as:

$$\|\mathbf{A}\|_2 = \lambda_{max}(\mathbf{A}) \tag{3.83}$$

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |A_{i,j}|^2 \right)^{(1/2)} \tag{3.84}$$

The following inequalities from [178] are also useful:

$$\|\mathbf{A}\|_F \leq \sqrt{\text{rank}(\mathbf{A})} \|\mathbf{A}\|_2 \tag{3.85}$$

$$\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F \tag{3.86}$$

To generate a random field as in equation (3.49), a Cholesky decomposition followed by a matrix-vector multiplication is required. Let $\hat{Z} = \hat{\mathbf{L}}a + \mu$ be the computed random field sample in finite precision arithmetic. The roundoff

error for the addition of μ is of size u and negligible and will be ignored for the remainder of the error analysis.

3.4.2.2 Random field sample error bounds covariance matrix decomposition

The error associated with the Cholesky decomposition is considered. First, the case of a positive definite $\mathbf{C}$ is analysed. The backward error is given by $\Delta\mathbf{C}$ where:

$$\hat{\mathbf{L}}\hat{\mathbf{L}}^T = \mathbf{C} + \Delta\mathbf{C}$$

From §10.1.1 of [178]:

$$\|\Delta\mathbf{C}\|_2 \leq \mathcal{O}(n^2u)\|\mathbf{C}\|_2 + \mathcal{O}(u^2) \tag{3.87}$$

By equation (3.46), an estimate for $\|\mathbf{C}\|_2$ is:

$$\|\mathbf{C}\|_2 \lesssim \sigma^2$$

For a stationary random field, $\sigma^2\mathbf{R} = \mathbf{C}$. The Cholesky decomposition can be calculated for $\mathbf{R}$ and then each of the independent degrees of freedom scaled by σ. By ordering the calculations this way, the forward error estimates can be reduced. To demonstrate this, calculations of bounds for both $\mathbf{R}$ and $\mathbf{C}$ are included. An estimate for $\|\mathbf{R}\|_2$ is:

$$\|\mathbf{R}\|_2 \lesssim 1$$

From §4.2.6 of [148], $\|\mathbf{L}\|_2^2 = \|\mathbf{C}\|_2$ so:

$$\|\mathbf{L}\|_2 \lesssim \sigma \tag{3.88}$$

For a stationary random field, if the decomposition is done using $\mathbf{R} = \mathbf{L}_\mathbf{R}\mathbf{L}_\mathbf{R}^T$, then:

$$\|\mathbf{L}_\mathbf{R}\|_2 \lesssim 1 \tag{3.89}$$

The backward error in $\mathbf{C}$ can be estimated from the above equations. Additionally, the result below holds for all (not just positive definite) $\mathbf{C}$ if the algorithms in [112] are adopted. The backward error in $\mathbf{C}$ and $\mathbf{R}$ is then:

$$\|\Delta\mathbf{C}\|_2 \lesssim \mathcal{O}(n^2u)\sigma^2 \tag{3.90}$$

$$\|\Delta\mathbf{R}\|_2 \lesssim \mathcal{O}(n^2u) \tag{3.91}$$

Combining equations (3.85), (3.90) and (3.91):

$$\|\Delta\mathbf{C}\|_F \lesssim \mathcal{O}(n^{(3/2)})u\sigma^2$$
$$\|\Delta\mathbf{R}\|_F \lesssim \mathcal{O}(n^{(3/2)})u$$

This gives the following useful result:

$$\frac{\|\Delta\mathbf{C}\|_F}{\|\mathbf{C}\|_2} \lesssim \mathcal{O}(n^{(3/2)})u \tag{3.92}$$

The forward error of the Cholesky decomposition is given by $\Delta\mathbf{L} = \mathbf{L} - \hat{\mathbf{L}}$. From Theorem 10.8 of [178] and [65]:

$$\frac{\|\Delta\mathbf{L}\|_F}{\|\mathbf{L}\|_2} \leq \mathcal{O}\left(\kappa_2(\mathbf{C})\right)\frac{\|\Delta\mathbf{C}\|_F}{\|\mathbf{C}\|_2} \tag{3.93}$$

$$\tag{3.94}$$

where $\kappa_2(\mathbf{C})$ is the condition number of $\mathbf{C}$. A large condition number represents an ill-conditioned system. Let $\lambda_{max}(\mathbf{C})$ and $\lambda_{min}(\mathbf{C})$ be the maximum and minimum eigenvalues of $\mathbf{C}$ respectively.

From §4.2.6 of [148], as $\mathbf{C}$ is symmetric positive definite:

$$\kappa_2(\mathbf{C}) = \frac{\lambda_{max}(\mathbf{C})}{\lambda_{min}(\mathbf{C})} \tag{3.95}$$

Bounds for $\kappa_2(\mathbf{C})$ can be derived if a rank-revealing decomposition, as discussed in Section 3.3.1.4, is used. Then, from §10.1.1 of [178], the decomposition will succeed if $20n^{(3/2)}\kappa_2(\mathbf{C})u \leq 1$. Then, for the linearly independent

part $\mathbf{C}_{IU}$:

$$\kappa_2(\mathbf{C}_{IU}) \leq \frac{1}{20n^{(3/2)}u} \tag{3.96}$$

This gives a lower bound estimate for $\lambda_{min}(\mathbf{C}_{IU})$:

$$\lambda_{min}(\mathbf{C}_{IU}) \gtrsim \mathcal{O}(n^{(3/2)}u)\sigma^2 \tag{3.97}$$

or for a stationary random field:

$$\lambda_{min}(\mathbf{R}_{IU}) \gtrsim \mathcal{O}(n^{(3/2)}u) \tag{3.98}$$

The error bound in equation (3.93) is affected by the size of $\kappa_2(\mathbf{C})$ and may be as bad as $\mathcal{O}(\kappa_2(\mathbf{C})u)$ for a general symmetric positive definite matrix [178].

An improved error bound estimate can be made investigating the structure of covariance matrices. From equation (3.50), the correlation between two random degrees of freedom, represented by vectors x and y, in a random field is given by the relative angle between them. The angle between two independent degrees of freedom is at most $\pi/2 \approx \mathcal{O}(1)$. As the x and y become co-linear, the angle between them approaches zero. Additionally, as x and y become co-linear $\mathbf{C}$ approaches singularity and one of the eigenvalues of the system approaches 0.

From [249], the equation of the ellipse described by two column vectors L_i and L_j of $\mathbf{L}_{\mathbf{R}IU}$, with correlation $\rho = \cos\theta$ as in equation (3.50) is:

$$\begin{aligned} x_1^2 - 2\rho x_1 x_2 + x_2^2 &= \left(1 - \rho^2\right) \\ x_1^2 - 2\cos\theta x_1 x_2 + x_2^2 &= \sin^2\theta \end{aligned} \tag{3.99}$$

where x_1 and x_2 are canonical basis vectors. Let A be the area of the ellipse given by equation (3.99). For the implicit equation of an ellipse in the form

$a_1x_1^2 + a_2x_1x_2 + a_3x_2^2 = 1$, the area of the ellipse is:

$$\begin{aligned}
A &= \frac{2\pi}{\sqrt{4a_1a_3 - a_2^2}} \\
A &= \frac{2\pi}{\sqrt{4\left(\frac{1}{\sin^2\theta}\right)^2 - \left(\frac{-2\cos\theta}{\sin^2\theta}\right)^2}} \\
A &= \frac{2\pi}{\sqrt{\left(\frac{4}{\sin^4\theta}\right)(1 - \cos^2\theta)}} \\
A &= \frac{2\pi}{\sqrt{\frac{4}{\sin^2\theta}}} \\
A &= \pi\sin\theta
\end{aligned} \tag{3.100}$$

The area of an ellipse is also, however, given by $A = \pi ab$ where a and b are the lengths of the semi-major axes of the ellipse. From [129] and the discussion in Section 3.3.1.2, for $\mathbf{R}_{IU}$, the length of semi-major axes of the ellipse are equal to $\sqrt{\lambda_1}$ and $\sqrt{\lambda_2}$ where λ_1 and λ_2 are the eigenvalues corresponding to the random degrees of freedom that represent elements i and j. Then:

$$\sin\theta = \sqrt{\lambda_1\lambda_2} \tag{3.101}$$

The eccentricity of an ellipse with semi-major axis lengths a and b with $b \leq a$ is defined as $e = \sqrt{1 - b^2/a^2}$. By definition, $e = 0$ for a circle and $e < 1$ for an ellipse. So when $\lambda_1 = \lambda_2$, $\sin\theta = 0$. Then as θ increases, $\sqrt{\lambda_1\lambda_2}$ must also increase. As θ approaches the maximum of $\pi/2$, $\sin\theta$ approaches 1. Then increasing λ_1 must decrease λ_2 and vice versa. From equations (3.44) to (3.46), $\lambda_{max}(\mathbf{R}_{IU})$ and $\lambda_{min}(\mathbf{R}_{IU})$ are bounded between $\lambda_{min}(\mathbf{R}_{IU})$ and 1. Fixing $\lambda_1 = \lambda_{max}(\mathbf{R}_{IU})$:

$$\theta \to \frac{\pi}{2},\ \lambda_2 \to \lambda_{min}(\mathbf{R}_{IU}) \tag{3.102}$$

Let ψ_{min} be the minimum angle, as in equation (3.50), between two correlated degrees of freedom represented by vectors x and y. Then the maximum error,

δ, in the distance of y from x is:

$$\begin{aligned} \delta &\approx \alpha \sin \psi_{min} \\ \delta &\approx \alpha \mathcal{O}(\psi_{min}) \end{aligned} \tag{3.103}$$

where α is the distance moved along y. So:

$$\sin \psi_{min} \approx \sqrt{\lambda_{max}(\mathbf{R}_{IU}) \lambda_{min}(\mathbf{R}_{IU})} \tag{3.104}$$

In standard IEEE 754 floating point arithmetic, there is no loss of precision for the square root operation [196]. Then, if the lower bound in equation (3.98) is satisfied, the precision of $\sqrt{\lambda_{min}}$ is roughly the same as that of λ_{min}:

$$\sqrt{\lambda_{min}} \approx \mathcal{O}(n^{(3/2)}u) \tag{3.105}$$

Combining equations (3.104) and (3.105):

$$\mathcal{O}(\psi_{min}) \approx \alpha \mathcal{O}(n^{(3/2)}u) \tag{3.106}$$

From equation (3.54), a random field is generated by multiplying $\mathbf{L}_{\mathbf{R}IU}$ by a vector, a, of n samples from a standard normal distribution. Each value in the final random field is then generated by moving a distance a_i along at most n of the random degrees of freedom described by the row vectors of $\mathbf{L}_{\mathbf{R}IU}$. From equation (3.106), the error in each of these n multiplications is:

$$a_i \mathcal{O}(n^{(3/2)}u) \tag{3.107}$$

To make use of equation (3.107), an estimate for the magnitude of each a_i is needed. Technically each a_i is unbounded as it is a sample from a standard normal distribution, and therefore the theoretical upper bound of $|a_i|$ is infinite. However, it would be exceedingly unlikely for each of the n entries in a to be much more than ± 3 and certainly less than say ± 100. Then, a practical estimate for $|a_i|$ is:

$$|a_i| \lesssim \mathcal{O}(10) \tag{3.108}$$

Note that the result of the multiplication $\mathbf{L_R}a$ must be increased by a factor of σ to scale the random field to the correct standard deviation. By §2.7.8 of [148], the error associated with scalar multiplication is approximately $\mathcal{O}(u\sigma\|\mathbf{L_R}\|) \approx \mathcal{O}(u\sigma)$. Then, for a stationary random field, decomposing $\mathbf{R}$ and multiplying $\mathbf{L}_{\mathbf{R}IU}a$ by σ only scales up the error in $\mathbf{L}_{\mathbf{R}IU}a$ by approximately σ. Then the roundoff error, $\Delta Z = Z - \hat{Z}$, for entry, $Z(i)$, in the random field sample is approximately:

$$\begin{aligned} \Delta\hat{Z}(i) &\lesssim 10\sigma n\mathcal{O}(n^{(3/2)}u) \\ \Delta\hat{Z}(i) &\lesssim \sigma\mathcal{O}(10n^{(5/2)}u) \end{aligned} \tag{3.109}$$

For a singular correlation matrix, if the procedure in Section 3.3.1.4 is adopted, all random degrees of freedom separated by less than ψ_{min} are considered co-linear. Then, by a similar argument above, the maximum angular error between two degrees of freedom, r and s, is ψ_{min}. If the angular error were more than this, then the λ_2 for the ellipse representing r and s would be larger than $\lambda_{min}\mathbf{L}_{\mathbf{R}IU}$ and r and s would not be numerically co-linear. Then if the angular error for assumed co-linear degrees of freedom is at most ψ_{min}, then the same error given by equation (3.109) holds.

Finally, the magnitude of the variance reduction γ needs to be considered. The actual magnitude of the minimum eigenvalue of $\mathbf{R}$ is:

$$\lambda_{min}(\mathbf{R}_{IU}) \approx \frac{\mathcal{O}(n^{(3/2)}u)}{\gamma_{min}} \tag{3.110}$$

If the magnitude of γ_{min} is greater than approximately 0.5, the magnitude of the error in $\lambda_{min}(\mathbf{R}_{IU})$ is unchanged. So then to maintain the precision estimates given in this Section, the following bound on the minimum variance reduction should be adopted as:

$$\gamma_{min} \geq 0.5 \tag{3.111}$$

From equation (3.109) and u as in equation (3.81), the rounding error can be

estimated for different n:

$$\text{For } n \approx 10^3,\ |\Delta\hat{Z}| \lesssim 10^{-8}\sigma$$
$$\text{For } n \approx 10^4,\ |\Delta\hat{Z}| \lesssim 10^{-5}\sigma$$
$$\text{For } n \approx 10^5,\ |\Delta\hat{Z}| \lesssim 10^{-3}\sigma$$

Note that these error estimates are very rough and represent a worst case scenario without any attempt to alleviate rounding errors by, say, iterative refinement as in [178, 148].

3.4.2.3 Rounding errors in FEM

To complete the error analysis of the covariance matrix decomposition random field generator, the rounding errors of FEM are discussed. Aside from measurement errors in the inputs, there are several sources of error in FEM. These sources of error include the fineness of the mesh (h-refinement), degree of the approximating polynomials (p-refinement) and the machine precision unit round off u.

Error estimates for the maximum theoretical precision of FEM, sufficient for the approximate bounds in this Chapter, are given in §6.8.3 of [377]. Although increasing h improves the accuracy of the computed FEM solution as $\mathcal{O}(h^{-2})$, eventually rounding errors prevent finer mesh elements from improving accuracy. Let q be the order of the finite element approximating polynomial. Then the maximum theoretical precision of FEM, δ is approximately:

$$\delta \approx \mathcal{O}(u^{(q+1)/(q+3)}) \tag{3.112}$$

Then, from [377], the estimates for the best case precision for FEM are:

$$\text{For } q = 1,\ \delta \approx 10^{-8}$$
$$\text{For } q = 2,\ \delta \approx 10^{-10}$$
$$\text{For } q = 3,\ \delta \approx 10^{-11}$$

Further, for implicit plastic finite element analysis, some form of residual convergence tolerance must be specified. The rounding errors in plastic FEM are

at least as large as the specified tolerance [54].

3.4.3 Discussion

The error and computational complexity characteristics of RFEM, MCS based Uncertainty Quantification using an FEM solver representation and covariance matrix decomposition as a random field generator, were discussed. The computational complexity of the repeated FEM iterations were found to be the computationally limiting factor for RFEM, not the random field generator. Also, bounds on the rounding errors for FEM and Cholesky decomposition were discussed. The rounding errors for Cholesky decomposition, without error reducing techniques, may become problematic for 10^4 or 10^5 elements. However, for this many elements, an RFEM analysis, particularly for a high probability of failure, will be very time consuming. For a specific problem, the bounds presented in this Chapter may help to estimate the approximate size of a feasible solution compared to the rounding errors. It is also worth noting that the measurement accuracy for many Civil Engineering problems are likely to be worse than any rounding errors for a reasonable problem. Additionally, improving the p order of the finite element approximation allows for the number of elements to be reduced, which would lessen errors in the random field sample.

3.5 Conditional random fields

Conditional random fields can be used to model the effect of observation data on random fields. Conditional models can also be used to fit probabilistic models to spatial data by calculating model likelihood terms. These techniques, along with a numerical demonstration, are presented in this Section. In the context of Machine Learning and Bayesian statistics, Gaussian random fields are typically referred to as Gaussian processes and are used to model probability distributions over functions, F. That is, given data $\mathcal{T}$, a distribution over a space of functions given data, $P(f|\mathcal{T})$, is modelled. The desired predictive distribution is then given by:

$$P(y^*|x^*, \mathcal{T}) = \int_F P(y^*|f, x^*)P(f|\mathcal{T})df \tag{3.113}$$

Using Bayesian updating, observed data about the values of a function can be incorporated with prior beliefs over a function space into a posterior distribution over functions. Viewed in this way, conditional random fields are non-parametric probability density models. A Gaussian process is a particular model of distributions over functions that assumes (as per the discussion in Section 3.2) that any collection of function values has a multivariate normal distribution.

3.5.1 Gaussian Process prediction and regression

Following from the more formal definition in Section 3.2, in the notation used by [308] and §15 of [275], the Gaussian process model of a function, $f(x)$, is denoted:

$$f(x) \sim GP\left(m(x), \kappa(x, x')\right) \tag{3.114}$$

$$\mu(x) = \mathbb{E}\left[f(x)\right] \tag{3.115}$$

$$\kappa(x, x') = \mathbb{E}\left[(f(x) - m(x))(f(x') - m(x'))^T\right] \tag{3.116}$$

where $m(x)$ is the mean function and $\kappa(x, x')$ is the covariance (positive definite) function between the points x, x'.

For any finite collection of N points, $\{x_i\}_{i=1}^N = X$, the Gaussian process model says that the function values are jointly normal:

$$P(f|X) = \mathcal{N}(f|\mu, K) \tag{3.117}$$

where $\mu = (m(x_1), \cdots, m(x_N))$ is a mean vector of a Gaussian with covariance matrix K with entries $K_{ij} = (\kappa(x_i, x_j))$.

Of interest is the posterior distribution of the Gaussian process given new data. Observing new data (values of the function to be inferred) reduces the variability over the function space and thereby restricts the forms of the functions that are considered likely. Specifically, high probability functions should be those that predict the data values with high probability. The Bayesian updating procedure for the posterior distribution is also known as Kriging in the geostatistics literature and conditional random field simulation (see [229]).

First, consider the noise free case. Let $X = \mathcal{T}$ be a training set of values $\{(x_i, f(x_i)\}_{i=1}^N$ where $f(x_i)$ is the value of the unknown function at x_i. Given a test set of points $X^* \in \mathbb{R}^{N^* \times D}$ where D is the dimension of each x_i, the goal is to predict the outputs of the function f^* at the test points. From the definition of a Gaussian process, any collection of points is jointly Gaussian. Then, the joint distribution over the training data and the test points is:

$$\begin{bmatrix} f \\ f^* \end{bmatrix} \sim P(f^*, f|X, X^*) = \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu^* \end{bmatrix}, \begin{bmatrix} K & K^* \\ K^{*,T} & K^{**} \end{bmatrix}\right) \tag{3.118}$$

where the covariance matrix blocks are given by:

$$K_{ij} = \kappa(x_i, x_j) \qquad x_i, x_j \in X; K \in \mathbb{R}^{N \times N} \tag{3.119}$$

$$K^*_{ij} = \kappa(x_i, x^*_j) \qquad x_i \in X; x^*_i \in X^*; K^* \in \mathbb{R}^{N \times N^*} \tag{3.120}$$

$$K^{**}_{ij} = \kappa(x^*_i, x^*_j) \qquad x^*_i, x^*_j \in X^*; K^{**} \in \mathbb{R}^{N^* \times N^*} \tag{3.121}$$

3.5.2 Bayes Rule for Gaussian Linear Systems

Before continuing, consider the *Gaussian Linear System* from §4.4 in [275]. Let $h \in \mathbb{R}^{D_h}$ be a vector of hidden variables and $v \in \mathbb{R}^{D_v}$ be a vector of noisy observations of h. Consider the Gaussian prior on h:

$$P(h) = \mathcal{N}(x|\mu_x, \Sigma_x) \tag{3.122}$$

and the associated likelihood $P(v|h)$:

$$P(v|h) = \mathcal{N}(v|Ah + b, \Sigma_v) \tag{3.123}$$

with the transformation matrix $A \in \mathbb{R}^{D_v \times D_h}$.

Then, the Bayes posterior update for h given v is given, from Theorem 4.4.1 of [275], as:

Theorem 3.5.1. *Bayes Rule for linear Gaussian Systems:* Consider a linear Gaussian system for $P(h)$ and $P(h|v)$, as described in equations (3.122) and

(3.123). Then, the posterior $P(h|v)$ is:

$$P(h|v) = \mathcal{N}(h|\mu_{h|v}, \Sigma_{h|v}) \tag{3.124}$$

$$\Sigma_{h|v}^{-1} = \Sigma_h^{-1} + A^T \Sigma_v^{-1} A \tag{3.125}$$

$$\mu_{h|v} = \Sigma_{h|v} \left[A^T \Sigma_y^{-1} (v - b) + \Sigma_h^{-1} \mu_h \right] \tag{3.126}$$

with normalisation $P(v)$:

$$P(v) = \mathcal{N}\left(v|A\mu_h + b, \Sigma_v + A\Sigma_h A^T\right) \tag{3.127}$$

Proof. See §4.3 of [275]. □

The proof computation relies on the equations for the marginal and conditional probabilities of the normal distribution. Consider a joint Gaussian distribution, $P(x_1, x_2)$, with marginals $P(x_1)$ and $P(x_2)$ and conditional distributions $P(x_2|x_1)$ and $P(x_1|x_2)$. The marginal and conditional distributions are also Gaussian. Following [308] and §4.3 [275], given a joint normal distribution:

$$P(x_1, x_2) = \mathcal{N}\left(\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right) \tag{3.128}$$

with inverse covariance matrix:

$$\Lambda = \Sigma^{-1} = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix} \tag{3.129}$$

has marginal distributions:

$$P(x_1) = \mathcal{N}(x_1|\mu_1, \Sigma_{11}) \tag{3.130}$$

$$P(x_2) = \mathcal{N}(x_2|\mu_2, \Sigma_{22}) \tag{3.131}$$

$$\tag{3.132}$$

and conditional distributions:

$$P(x_1|x_2) = \mathcal{N}(x_1|\mu_{1|2}, \Sigma_{1|2}) \tag{3.133}$$
$$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \tag{3.134}$$
$$= \mu_1 - \Lambda_{11}^{-1}\Lambda_{12}(x_2 - \mu_2) \tag{3.135}$$
$$= \Sigma_{1|2}(\Lambda_{11}\mu_1 - \Lambda_{12}(x_2 - \mu_2)) \tag{3.136}$$
$$\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} = \Lambda_{11}^{-1} \tag{3.137}$$

Note that $\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$ is termed the *Schur compliment* of Σ_{22} in Σ [148].

3.5.3 Conditional random field posterior

Using the formulas for conditional distributions of a joint Gaussian given in equations (3.133), (3.134) and (3.137), the posterior $P(f^*|X^*, X, f)$ is given by:

$$P(f^*|X^*, X, f) = \mathcal{N}(f^*|\mu_*, \Sigma^*) \tag{3.138}$$
$$\mu^* = \mu(X^*) + K^{*,T}K^{-1}(f - \mu(X)) \tag{3.139}$$
$$\Sigma^* = K^{**} - K^{*,T}K^{-1}K^* \tag{3.140}$$

For a full derivation of these equations, see §2 of [308] and §15.2.1 of [275].

When the observations of the true function are noisy and if the form of the noise is not complicated then this error can be incorporated into the Gaussian process estimate of $f(x)$. Let the noise estimate of $f(x)$ be given by:

$$\hat{f}(x) = f(x) + \epsilon \tag{3.141}$$

where $\epsilon \sim \mathcal{N}(0, \sigma_x^2)$. Then, the covariance is given by:

$$\begin{aligned}
\text{Cov}[x, x'] &= \mathbb{E}\left[(\hat{f}(x) - m(x))(\hat{f}(x') - m(x'))^T\right] \\
\text{Cov}[x, x'] &= \mathbb{E}\left[(f(x) + \sigma_x^2 - m(x))(f(x') + \sigma_{x'}^2 - m(x'))^T\right] \\
\text{Cov}[x, x'] &= \mathbb{E}\left[(f(x) - m(x))(f(x') - m(x'))^T\right] \\
&\quad + \mathbb{E}\left[\sigma_x^2(f(x') + \sigma_{x'}^2) + f(x)\sigma_{x'}^2 - m(x)\sigma_{x'}^2 - m(x')\sigma_x^2\right] \\
\text{Cov}[x, x'] &= \kappa(x, x') + \mathbb{E}\left[f(x)\sigma_{x'}^2 + f(x')\sigma_x^2 + \sigma_x^2\sigma_{x'}^2\right] \\
&\quad - m(x)\mathbb{E}\left[\sigma_{x'}^2\right] - m(x')\mathbb{E}\left[\sigma_x^2\right] \\
\text{Cov}[x, x'] &= \kappa(x, x') + \mathbb{E}\left[f(x)\sigma_{x'}^2\right] + \mathbb{E}\left[f(x')\sigma_x^2\right] + \mathbb{E}\left[\sigma_x^2\sigma_{x'}^2\right]
\end{aligned}$$

The simplest case is to assume that the noise is independent of the signal and that the noise is pointwise independent from itself so:

$$\mathbb{E}\left[f(x)\sigma_{x'}^2\right] = 0 \tag{3.142}$$

$$\mathbb{E}\left[f(x')\sigma_x^2\right] = 0 \tag{3.143}$$

$$\mathbb{E}\left[\sigma_x^2\sigma_{x'}^2\right] = \delta(x, x')\sigma_x\sigma_{x'} = \delta(x, x')\sigma_x^2 \tag{3.144}$$

so that the covariance is given by:

$$\text{Cov}[x, x'] = \kappa(x, x') + \delta(x, x')\sigma_x^2 \tag{3.145}$$

Other more complex models of the noise are discussed in §2 and §9 of [308].

Given the noise model in equation (3.145), the covariance matrix of N points from the Gaussian process is:

$$\text{Cov}[\hat{f}|X] = K + \sigma_x^2\mathbf{I}_N = K_\epsilon \tag{3.146}$$

so that the effect of the noisy estimates is to add a diagonal component to the covariance matrix.

To evaluate the posterior distribution using the noisy training data, the joint distribution in equation (3.118) is modified to use K_ϵ rather than K. Note that the mean of a Gaussian process can always be subtracted away and the process modelled by a zero mean Gaussian. The mean function can be added back on to the process for simulation. For notational convenience the zero mean formulation is given below. The modified joint distribution that incorporates

data noise, with zero mean, is given by:

$$\begin{bmatrix} \hat{f} \\ f^* \end{bmatrix} \sim P(f^*, \hat{f}|X, X^*) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_\epsilon & K^* \\ K^{*,T} & K^{**} \end{bmatrix} \right) \tag{3.147}$$

which yields the posterior distribution under noisy data:

$$P(f^*|X^*, X, f) = \mathcal{N}(f^*|\mu^*, \Sigma^*) \tag{3.148}$$

$$\mu^* = K^{*,T} K_\epsilon^{-1}(y) \tag{3.149}$$

$$\Sigma^* = K^{**} - K^{*,T} K^{-1} K_* \tag{3.150}$$

An important case is when the test point is a single location in the function domain, x^*. In this case, the posterior predictive distribution is, from §15.2.2 of [275], given by:

$$P(f^*|x^*, X, \hat{f}) = \mathcal{N}(f^*|k^{*,T} K_\epsilon^{-1} \hat{f}, k^{**} - k^{*,T} K_\epsilon^{-1} k^* \tag{3.151}$$

where k^* denotes the covariance operator between the test point and the data and k^{**} is the covariance operator at the test point:

$$k^* = [\kappa(x^*, x_1), \cdots, \kappa(x^*, x_N)] \tag{3.152}$$

$$k^{**} = \kappa(x^*, x^*) \tag{3.153}$$

The posterior predictive mean at a single test point is also often given by:

$$\mathbb{E}\left[f^*(x)\right] = k^{*,T} K_\epsilon^{-1} \hat{f} \tag{3.154}$$

$$= \sum_{i=1}^{N} \alpha_i \kappa(x_i, x^*) \tag{3.155}$$

where α_i are the entries vector α which is the product:

$$\alpha = K_\epsilon^{-1} \hat{f} \tag{3.156}$$

3.5.4 Model likelihood

Consider a parameterised representation of a covariance function, $\kappa(x, x'; \theta)$. One approach to estimate possible values for θ is to maximise the marginal

likelihood (that is, the model evidence) of the estimated function values given the data. Other methods for optimising the model parameters include Variational Bayesian methods [46]. Following §15.2.4 in [275] and [308], the goal is to estimate and maximise:

$$P(\hat{f}|X) = \int_F P(\hat{f}|f, X)P(f|X)df \tag{3.157}$$

$$P(\hat{f}|X) = \int_F P(\hat{f}, f|X)df \tag{3.158}$$

under the Gaussian process prior for $f(x)$ and under the assumption that the noisy estimate $\hat{f}(x) = f(x) + \epsilon$ has independent noise, as in equation (3.141), so that:

$$P(f|X) = \mathcal{N}(0, K) \tag{3.159}$$

$$P(\hat{f}|f) = \prod_{i=1}^{N} \mathcal{N}(y_i|f_i, \sigma^2) \tag{3.160}$$

From equation (3.146), the covariance of $\hat{f}$ given X is K_ϵ so then, assuming the measured values for $\hat{f}$ have had their estimated mean removed, the model evidence is given by:

$$P(\hat{f}|X) = \mathcal{N}(0, K_\epsilon) \tag{3.161}$$

As maximising the log of the marginal likelihood is equivalent to maximising the marginal likelihood, the objective is to maximise:

$$\log P(\hat{f}|X) = \mathcal{N}(0, K_\epsilon) \tag{3.162}$$

$$\log P(\hat{f}|X) = -\frac{1}{2}\hat{f}K_\epsilon^{-1}\hat{f} + \frac{1}{2}\det\left[K_\epsilon\right] - \frac{N}{2}\log(2\pi) \tag{3.163}$$

For further details on equation (3.163), see §15.2.4 of [275] and §2 of [308].

Consider a covariance function parameterised by the vector θ with components θ_i so that the likelihood in equation (3.163) is expressed as $\log P(\hat{f}|X;\theta)$. From [374], the gradient of the likelihood function with respect to θ_i has an analytical form:

$$\frac{\partial}{\partial\theta_i}\log P(\hat{f}|X;\theta) = \frac{1}{2}\hat{f}^T K_\epsilon^{-1}\frac{\partial K_\epsilon^{-1}}{\partial\theta_i}K_\epsilon^{-1}\hat{f} - \frac{1}{2}\operatorname{tr}\left(K_\epsilon^{-1}\frac{\partial K_\epsilon^{-1}}{\partial\theta_i}\right) \tag{3.164}$$

Although it is possible to perform gradient ascent on the likelihood by via the gradient in equation (3.164), there are two dual analytical solutions in the isotropic Gaussian noise case. Probabilistic Principal Components Analysis (see [374]) considers a regression type formulation of the latent function and analyses the principal components of the basis function regression weights. The Gaussian Process-Latent Variable Model (see [215]) performs an eigen-analysis on the empirical covariance matrix calculated directly with the observed function values. These methods can be used to estimate the model likelihood of a proposed Gaussian process model of spatial data. In this way, Conditional Random Field theory can be used fit models to data in preparation for additional probabilistic analysis.

3.5.5 Practical information

A variety of software is available for Conditional random field simulation. In particular, the Python library Scikit-learn [293] features a large number of Conditional Random Field examples, along with code to generate these examples, from a variety of data sets. Further, there are a number of computation saving techniques that may be more efficient in certain cases. Of particular note are classical approaches based on the Turning Bands Method as discussed in [205, 91].

For examples of applied conditional random field simulation in Civil Engineering contexts, see [382, 118, 201]. In [333], conditional simulations are used to estimate the effective permeability of heterogeneous media. The effective permeability is described by the geometric mean of a lognormal distribution, which is known to be the distribution's median [114].

3.6 Conclusions

This Chapter provided an overview of the theoretical and computational aspects of modelling spatially random fields. The intention of this overview is to enable the further developments in Uncertainty Quantification methods and probability theory presented in the subsequent Chapters of this thesis. Gaussian random fields were discussed with reference to the KL Expansion. Non-Gaussian random fields were discussed with reference copula theory. Phase

field models combining point processes with spatial processes were also detailed. Simulation techniques and examples for each of these methods were detailed. Further, numerical issues and the computational complexity of the simulation methods were analysed with reference to Uncertainty Quantification. The simulations presented were all shown in two dimensions. However, the techniques presented are not limited to two dimensional simulations. This choice was made to allow for illustrative Figures to be presented, rather than because of mathematical limitations.

Gaussian random field methods based on covariance matrix decomposition were discussed. Without significant modifications, however, the methods for spatially autocorrelated random field simulation presented are limited to Gaussian random fields, non-Gaussian fields with Gaussian copula, and other simple transformations of Gaussian random fields such lognormal random fields. The same is true of most random field simulation techniques [351] and the simulation of non-Gaussian random fields remains an active area of research. As such, this limitation of matrix decomposition is not considered to be major. Further, only simple phase field models were presented. Improving these techniques, with reference to the developments in [229], would be a worthwhile area of future research.

Conditional random field simulation, or Kriging, was also discussed in this Chapter. Conditional random field simulation can be used to generate random samples with fixed (or approximately fixed) values at certain locations. By considering the form of the posterior distribution after observing data, conditional random fields can also be used to fit models to data (for example selecting an appropriate correlation function) by calculating model likelihoods. Parameter estimation for random field models from data was not discussed extensively. Conditional random field models can be used to fit Gaussian Process models by estimating model likelihood values. Further discussion is presented in [118, 21, 275]. The developments in this Chapter are a sufficient foundation from which to discuss the Uncertainty Quantification of physical systems presented analyses in subsequent Chapters.

Chapter 4

Efficient Markov Chain Monte Carlo for combined Subset Simulation and Nonlinear finite element analysis

Contents

Figures

Tables

Chapter 4 Overview

Key developments in Chapter 4 include:

- Section 4.2.5 details background material on MCMC techniques suitable for use with a Subset Simulation method for rare event estimation using random, nonlinear Finite Element models.
- Section 4.2.7 presents an original contribution. Specifically, a rigorous confidence interval based convergence estimator for analysing the accuracy of Subset Simulation estimates of rare events is detailed. The confidence interval estimator is derived based on the product probability distribution of Monte Carlo Gaussian estimates implied by the Central Limit Theorem.
- Section 4.2.8 presents applications of the theoretical developments in this Chapter. The relative efficiency of different MCMC techniques are compared.
- Section 4.3.3 contributes new empirical results showing that the estimation accuracy for Subset Simulation on a probabilistic linear elastic problem can exceed both the Spectral Stochastic Finite Element Method and direct Monte Carlo Simulation.
- Section 4.3.4 contributes an analysis of the computational efficiency of Subset Simulation on a probabilistic test case problem: a footing on a elastoplastic soil. Subset Simulation is shown to be more computationally efficiency than direct Monte Carlo for rare event reliability estimation.

4.1 Introduction

This Chapter explores the application of Subset Simulation, a sampling based numerical integration technique, to the probabilistic analysis of structural models. Subset Simulation was originally presented in [15] for the estimation of rare event probabilities. Subsequent developments are detailed in [16]. In Civil Engineering, structural problems often have very small probabilities of failure but very large consequences of failure. The efficient estimation of these small probabilities would help to improve risk based estimates of structural safety. A useful definition of risk that can be adopted is probability times the consequence of an event [118]. By quantifying the risk associated with a given design, the level of risk can be assessed based on a comparison with other accepted risks [95]. For risk based analysis of consequential rare events the potential severity of an outcome may still carry high associated risks. In this scenario, it becomes necessary to estimate rare event probabilities for structural systems.

In soils, for example, spatial autocorrelation of material property parameters is known to have a significant impact on calculated failure probabilities [270]. Probabilistic analyses that aim to quantify risks accurately in such systems must then incorporate the effects of spatial autocorrelation into physical models. Ideally, spatial autocorrelation should be incorporated into existing advanced numerical models, allowing for the developments in probabilistic techniques and general numerical modelling of physical phenomena to occur in parallel. This Chapter uses spatially autocorrelated models of material property parameters for the physical models discussed.

Popular existing methods for calculating failure probabilities when random fields are used to define the input distribution are the Random Finite Element Method (RFEM) and the variants of the Stochastic Finite Element Method (SFEM). The original RFEM uses a direct Monte Carlo Simulation (MCS) procedure which is discussed further in Section 4.2.3 of this Chapter. SFEM and variants such as the Spectral SFEM (SSFEM) [141, 142] perform an integration over the random dimensions of the input probability space by expressing this space in terms of orthogonal polynomials (termed the Polynomial Chaos). The disadvantage of the SFEM style approach is that it is maximally inaccurate on the tails of the distribution [356] and as such is a poor candidate

for estimating small threshold probabilities very far from the mean response of the model output distribution. Additionally, the size of the probabilistic linear systems that must be solved during an SFEM analysis increases very rapidly with the dimensionality of the problem [141]. The convergence rate of Monte Carlo Simulation, by contrast, is independent of the dimensionality of the problem. Further, elastoplastic and other nonlinear material models are of interest in engineering practice. Unfortunately, using such models in SFEM type analyses is challenging and remains an active area of research, see [203, 334].

For sampling-type solutions of stochastic Partial Differential Equations (PDE), the nonlinearity of constitutive models can be accommodated easily as long as deterministic solutions for that material model are available. RFEM, a direct MCS method [118], is a powerful way of incorporating material property spatial autocorrelation into existing numerical models. RFEM proceeds by first selecting a random field model of material property parameters and/or applied loads, then repeatedly sampling random field realisations from the input distribution, running the deterministic inputs through a finite element solver and then using the output from the discrete deterministic problems to estimate the distribution of outputs. For reliability analysis, RFEM is a very useful approach because no specific assumptions are made on the output distribution (although this carries some disadvantages if the form of the output distribution is known a priori) and as such the full system reliability can be calculated regardless of the complexity of the input distribution.

Reliability analysis for Civil Engineering problems often involves the estimation of vanishingly small probabilities of failure [15, 16, 270]. Direct MCS is theoretically able to compute rare event probabilities, but may require a prohibitively large number of discrete simulations to do so accurately [336, 15]. For an RFEM approach, the dimensionality of the probabilistic input space is on the order of the number of elements in a mesh needed to approximate the discrete, deterministic simulations accurately. In the case that nonlinear PDE problems must be solved for each Monte Carlo iteration, the computational cost of random field simulation is negligible compared to the cost of the PDE problems. This issue is discussed in Section 4.2 of this Chapter. In this case, the particular choice of random field simulator is not nearly as significant as minimising the number of Monte Carlo iterations. Direct MCS is poorly suited

to estimating rare event probabilities associated with numerical approximations to nonlinear PDE systems because the number of iterations can not be reduced without sacrificing accuracy.

The underlying problem for rare event simulation with combined finite element and random field models is the difficulty of search in high dimensional spaces. As high dimensional spaces are very "large" volumetrically, it can take a long time to find rare portions of the search space by a stochastic search. Human intuition, which is well suited for explorations of one to three dimensional spaces, is poorly adapted to high dimensional search [6]. Further, as the number of probabilistic dimensions increases, it becomes increasingly time consuming to compute quantities of interest accurately and quickly. Techniques that are successful in a small number of dimensions may fail in very high dimensional spaces, as is the case with SFEM. The dimensionality of the numerical problem to be solved in an SFEM type analysis grows by the factorial of the approximation order [356].

To address the small probability of failure problem, Subset Simulation can be applied [15, 16]. Subset Simulation, also known as Sequential Monte Carlo, has been applied in areas other than probabilistic engineering mechanics [169, 99]. This method is able to calculate small threshold event probabilities more efficiently than direct MCS and is discussed in more detail in Section 4.2.4. Subset Simulation proceeds by performing a series of Markov Chain Monte Carlo analyses. Essentially, these analyses estimate the probability of some lesser failure condition than the full, desired failure condition. For example, if failure is deemed to occur once displacements of a system exceed some threshold value, a subset failure event would be perhaps one in which displacements reach 80% of the failure threshold level. Subset Simulation has previously been combined with RFEM and applied to slope stability problems [329, 239, 238]. These publications demonstrate that the Subset Simulation approach is able to capture small probabilities of failure for finite element systems with material property parameters modelled by spatially autocorrelated random fields. The componentwise Markov Chain Monte Carlo approach in [329, 239], however, can be improved on by using a more appropriate sampling methodology. These improvements are the subject of this Chapter.

This Chapter seeks to address some of the known issues (see [290, 55]) related to Markov Chain Monte Carlo in high dimensional spaces. In particular, this

Chapter compares the performance of Metropolis-Hastings (MH), Gibbs and Componentwise Metropolis Hastings (CMH) sampling. CMH is sometimes referred to as Modified-Metropolis or Metropolis-in-Gibbs sampling. In this Chapter, the term Componentwise Metropolis Hastings is preferred for the reasons outlined in [137]. Metropolis-Hastings generates new samples based on the previous random sample from a distribution. In high dimensional spaces, Metropolis-Hastings faces both numerical stability issues and problems with disappearing sample acceptance rates [16]. Both Gibbs and CMH sampling attempt to avoid these issues by adjusting only a single probabilistic degree of freedom per iteration [137]. CMH, in particular, is the favoured approach in the Subset Simulation literature [15, 16, 239, 329]. The justification for adopting this sampling technique is that the rejection rate for MCMC random walk in high dimensional spaces is too low to allow for effective traversal of the search space [16, 290]. Higher acceptance rates can be achieved by adjusting only a single random dimension in the random field per Monte Carlo iteration. Componentwise random walk MCMC, however, requires a larger number of discrete simulations to be carried to converge on a target probability estimate [55]. This is not desirable for probabilistic FEM analyses or any other case in which sampling is computationally expensive.

This Chapter directly compares different MCMC sampling methodologies for Subset Simulation. The theoretical developments detailed in Section 4.2 are verified by two test case analyses. The first numerical analysis is a verification study that compares the performance of Subset Simulation using MH, CMH and Gibbs sampling to MCS and SSFEM on the linear elastic footing problem originally presented in [359, 360]. The second numerical analysis is a comparative study of the performance of Subset Simulation (for MH, CMH and Gibbs sampling) and MCS for a footing on elastoplastic soil. General details on elastoplasticity in geomaterials are provided in [70, 96, 88, 299]. For the comparative study, the footing problem from the verification study is modified and extended. The parameters of the material model were selected to be similar to real world values and also to intentionally create a small probability for far from mean responses to allow for a comparison between Subset Simulation and MCS. Numerical results are presented in Section 4.3.

The performance of Subset Simulation is tested by considering the relative performance of Gibbs sampling and Componentwise Metropolis-Hastings sam-

pling (as used in [239, 16]) to direct Metropolis-Hastings sampling which has been adjusted to allow for stable numerical computation in high dimensional spaces. The numerical analysis indicated that for both linear and nonlinear finite element problems, Metropolis-Hastings outperformed (in the sense of computational effort versus accuracy) both Gibbs and CMH sampling. §4.5 of [16] argues that the acceptance rate for state transitions in Metropolis-Hastings tends to zero as the size of the random vector increases and so it becomes very difficult to locate a sample that is accepted. As such, [16] recommend adopting componentwise techniques, in particular CMH. However, in the context of nonlinear finite element analysis, the rejection of samples in the MCMC phase is not nearly as significant a computation problem as the solution of the discrete finite element problems. The vast difference in computational effort between nonlinear FEM and MCMC random step search means that it is preferable to minimise the number of FEM solutions computed by tolerating a higher search time for the next MCMC step rather than jumping more frequently between less significant parts of the stationary distribution. In the event that a random step is not made to a new sample, the FEM equations do not need to be re-evaluated and as such the search for the next MCMC sample can proceed rapidly.

This Chapter also presents a new derivation of confidence interval error estimation for Subset Simulation. The typical approach to computing confidence intervals for Subset Simulation is to run repeated Markov Chain analyses to bound the output response range [15]. When sampling is computationally expensive (as is the case for Subset Simulation combined with nonlinear FEM), multiple repeats of Markov Chains is an extremely inefficient method for generating reliability expectation range estimates. This Chapter presents a technique for efficiently estimating Subset Simulation confidence intervals directly from an analytic integral formula, rendering the estimation of these intervals computationally tractable for any Subset Simulation problem.

The Central Limit Theorem allows for the error for either MCS or MCMC to be given in terms of a normal distribution. The product of normal distributions is, however, not a simple distribution. The variance of the product distribution can be calculated directly from a simple formula and then used to provide standard error estimates [16]. The calculation of confidence intervals, however, requires that integrals of the product distribution function can be

evaluated. Moreover, the product distribution is asymmetrical which creates further difficulties when estimating confidence intervals. The derivation of the product distribution mass function for approximation of Subset Simulation confidence intervals is presented in Section 4.2.7 along with a description of numerical techniques for estimating these confidence intervals.

4.2 Sampling methodologies for reliability analysis

4.2.1 Introduction

The goal of numerical reliability analysis is to quantify the distribution of possible outputs from a particular physical model. In this Chapter, the focus is on quantifying probabilities associated to threshold events. An example of a threshold event would be the probability that the displacements of a numerical model exceed some value. Given a mathematical model of a physical system, §2 in [356] identifies the sources of uncertainty in the system outputs as material property parameters, loading, boundary conditions and the problem geometry. These sources of uncertainty do not address the approximation error of the mathematical model compared to physical phenomena, which is discussed in [94]. Numerical reliability analysis allows for the outputs from a particular numerical model to be quantified, but practitioner judgement is still required in selecting a reasonable mathematical approximation to real world physical systems. Although minimising the discrepancy between mathematical and probabilistic models remains challenging, improving techniques for probabilistic analysis is still a critical part of engineering practice. Improving probabilistic models allows for better comparison between model and real world data, facilitating future improvements in probabilistic analysis. This Chapter will focus on reliability analysis for physical problems modelled as stochastic PDE's.

Adopting modified notation from §3 in [244] and [141], the reliability analysis problem can be stated as follows. Let $(\Omega, \mathcal{A}, P)$ be a probability space where $\omega \in \Omega$ are the input configurations, $\mathcal{A}$ is the σ-algebra generated by Ω and P is the probability measure, see [168] for the definitions of these terms. Assume that a physical model is defined over a domain D with points $x \in D$. Let the stochastic PDE representing some physical process over this domain can at

time t be represented as:

$$L(x,t,\omega)u(x,t,\omega) = f(x,t,\omega) \tag{4.1}$$

In the remainder of this Chapter, the explicit time dependence in equation (4.1) will be left out, as the focus of the numerical analyses in Section 4.3 will be on steady state problems. $L(x,\omega)$ represents some differential operator defined at each point $x \in D$ with source terms $f(x,\omega)$ and solutions $u(x,\omega)$.

The reliability analysis problem can be stated as [244]:

$$\overline{\Psi} \approx \mathbb{E}\left[\Psi(u_a(\cdot))\right] = \int_\Omega \Psi(u_a(\omega))P(d\omega) \tag{4.2}$$

where $P(d\omega)$ is the probability to have sampled ω, $u_a(\omega)$ is the numerical approximation of the solution to equation (4.1) for a given input sample ω. Then, let the threshold event of interest for reliability analysis be given by $\Psi(u(\omega))$ as follows:

$$\Psi(u_a(\omega)) = \chi_T = \begin{cases} 1 \text{ if threshold event occurs for } u_a(\omega) \\ 0 \text{ if threshold event does not occur for } u_a(\omega) \end{cases} \tag{4.3}$$

The expectation of the indicator function is equal to the probability of the indicated region, so $\mathbb{E}[\chi_T] = P(T)$. For example, in the reliability analysis of structural systems, a common Ψ would be a function which decides whether a given $u_a(\omega)$ counts as a failure or not.

Exact integration of the reliability integral in equation (4.2) is, almost always, completely intractable and so estimation techniques are required. If $\overline{\Psi}$ is vanishingly small, then standard integral approximation techniques can become computationally intractable. For example, quadrature based integration requires a rapidly increasing number of points to improve the tolerance of the solution of an integral problem [139]. Unmodified Monte Carlo based integration has an effectively fixed rate of convergence towards the solution of an integral problem but this convergence rate can be too slow to handle reliability problems with small $\overline{\Psi}$ in a reasonable amount of time. This Chapter presents extensions to a technique, Subset Simulation [15], for dealing with this issue when the space of probabilistic inputs is very high dimensional and the value of $\overline{\Psi}$ is very small.

4.2.2 Reliability problems with random field material parameter models

In this Chapter, $u_a(\cdot)$ is considered for PDE's representing physical systems. Further, only uncertainty in the material property parameters is considered. Material property uncertainty is modelled by a random field. The fully general definition of a real valued random field is given in Definition 1.1.1 in [5]. The multivariate Gaussian distribution with mean μ and correlation matrix Σ, denoted by $\mathcal{N}(\mu, \Sigma)$, can be used to represent a random field. For example, spatial correlation of material parameters within a physical domain can be captured in Σ. Numerical simulation of random fields is typically restricted to multivariate Gaussian distributions or simple transformations (for example multivariate log-normal) of Gaussian fields [118].

In this Chapter, as the focus is on the integration technique rather than on random field simulation, only Gaussian random fields are used. Although a Gaussian distribution is a potentially questionable choice as a model of soil properties [118], the numerical analyses presented in Section 4.3 are based on the problem originally analysed in [360]. The problem in [360] adopts a Gaussian random field as a model of soil properties. In order to perform the verification analyses in this Chapter against the results in [360], it was necessary to restrict the input model to a Gaussian random field. Further, the sampling integration techniques tested in this Chapter are independent of the method used to generate samples from a random field and as such could be extended to non-Gaussian random fields. Non-Gaussian random fields with specified marginal distributions and covariance structures can be simulated efficiently using the translation process concept, see [160], the methods described in [324, 296] and using the methods described in Chapter 3.

For the probabilistic problem in equation (4.2), the selection of a random field defines a probability distribution over the space of possible inputs to the function $u(\cdot)$. A random field defined over a d dimensional geometric domain, D in $\mathbb{R}^d$, is technically infinite dimensional. That is, there is a random variable for each point in the problem domain. To make random field simulation tractable, it is necessary to discretise the random field. There are two main methods that are used to discretise random fields for subsequent finite element analysis, local averaging and orthogonal series expansion methods. Local averaging simulation methodologies compute a single correlated random variable

for each element in the mesh. Series expansion random field simulation methods over FEM meshes typically compute a single random field parameter for each quadrature integral gauss point within each discrete element. For either case, let the number of discrete points at which the random field is to be approximated be given by M.

At a high level, random field simulation for finite element analysis proceed in essentially the same way. For both the locally averaged random field simulators in [118] and the Karhunen-Loève orthogonal series expansion simulators in [43], discrete realisations field are produced by generating a vector, γ of M independent, identically distributed (i.i.d.) standard normal samples. Then, a transformation, G is applied to γ that enforces the correlation structure between the random variables, so $G : \mathbb{R}^M \rightarrow \mathbb{R}^M$.

Consider the particular case:

$$G\gamma \mapsto \beta \tag{4.4}$$

The vector β contains a set of correlated samples from a standard normal distribution. To produce the desired output distribution, the β vector can be transformed to the correct mean and standard deviation. β could also be transformed from a Gaussian to another related distribution, such as a log normal distribution [118].

The general transformation G may be expressed in terms of matrix multiplication. The transformation matrix can be found by a variety of methods, for example, matrix decomposition of the covariance structure on either locally averaged elements [390, 158], between the Gauss points of an FEM mesh [43] or by some more general form of the Karhunen-Loève orthogonal series expansion [356, 351]. For Componentwise Metropolis-Hastings sampling all that is required of the random field sampling methodology is that the components of γ can be changed individually between subsequent random field simulations. As this procedure is very general it is not restricted to a particular random field simulator. Similarly, the direct Metropolis-Hastings procedure discussed in Section 4.2.5.5 adjusts the entire γ vector in a manner independent of the choice of the random field simulator represented by G.

The remainder of this section discusses techniques to estimate the integral in equation (4.2) in the case that the threshold event probability, $\overline{\Psi}$, is vanishingly small.

4.2.3 Integration by direct Monte Carlo Simulation

The direct Monte Carlo Simulation procedure for RFEM is briefly described in this Section. This is for both notational consistency and because a direct Monte Carlo stage may be used for the Subset Simulation procedure described in Section 4.2.6. In combination with finite element analysis, direct MCS has been used to model uncertain material property parameters, loading and geometry [118]. The Subset Sampling methodology extends MCS. Only material property uncertainty is modelled in the analyses presented in Section 4.3. The methodology presented could, however, be applied to loading and geometry uncertainty as long as discrete samples can be drawn from distributions modelling these random variables. Geometric uncertainty is discussed in [141, 325].

In very high dimensional spaces, the well known curse of dimensionality [13] prevents quadrature techniques from being used to compute the integral in equation (4.2). Let N be the number of points at which a function to be integrated is estimated at and M be the dimensionality of the problem. Then as M increases, the solution of the integral estimated by quadrature converges to the true solution like $O(N^{-1/M})$ (§11, [13]). For finite element problems with input distributions defined by random fields, the dimensionality of the inputs is at least M, which is the dimension of the approximation, Ω_a, to the random field Ω. For a locally averaged random field, M is on the order of the number of elements in the finite element mesh. Integration by quadrature is thus not viable for rare event random field/finite element analysis.

The simplest method to evaluate a high dimensional integral like equation (4.2) is Monte Carlo Simulation. From [118, 224], samples ω_a are taken uniformly at random from the approximation to the input distribution and $\Psi(u_a(\omega_a))$ solved for. Repeating this process, the reliability can be estimated by:

$$\overline{\Psi} = \int_{\Omega} \Psi(u_a(\omega)) P(d\omega) \approx \overline{\Psi}_N = \frac{1}{N} \sum_{i=1}^{N} \Psi(u_a(\omega_{a,i})) \tag{4.5}$$

Further, it can be shown that, as a consequence of the Central Limit Theorem (CLT), the convergence of the approximation in equation (4.5) is $O(N^{-1/2})$ [57]. Following [55], this is because as N becomes large, the CLT says that the $\overline{\Psi}_N$ approaches a Gaussian distribution if the samples $\Psi(u_a(\omega_a))$ are indepen-

dent and identically distributed (which is the case for Monte Carlo Simulation). Specifically, let σ^2 be the variance of $\Psi(u_a(\omega_a))$. Then as N approaches infinity, by the CLT, $\overline{\Psi}_N \approx \mathcal{N}\left(\overline{\Psi}, \frac{\sigma^2}{N}\right)$. Finally, 95% confidence intervals for probabilities estimated by MCS can then be calculated using $\overline{\Psi}_N \pm 1.96\frac{\sigma_N}{\sqrt{N}}$ [55].

The convergence of equations (4.5) and its associated variance is independent of the dimensionality of the problem. Unfortunately, when attempting to estimate very small probabilities, the convergence rate of MCS is poor. As the convergence rate improves like $\frac{1}{\sqrt{N}}$, an additional significant figure of accuracy requires (in base ten) that the sample size is increased by a factor of $10^2 = 100$. For example, in a typical Civil Engineering design problem, the probability of failure may be as small as $\sim 10^{-2}$ to 10^{-5} (see §5, [21]). A probability of failure of $\times 10^{-2}$ would require on the order of $N = 10^6$ analyses to achieve an accurate probability estimate by RFEM. For example, the case study presented in [103] is considered to have a high probability of unacceptable performance (approximately 0.2×10^{-2}) for a typical problem and so a full analysis in practice may require the estimation of much smaller probabilities of failure.

4.2.4 Subset Simulation

Subset Simulation is a useful technique for estimating small probabilities by calculating a chain of more easily estimated conditional probabilities. Subset Simulation avoids the poor convergence rate of direct MCS by estimating the desired probability in terms of a series of conditional probabilities, each of which converges rapidly. The methodology given here is described at least as early as [15]. Subsequent related works and developments of Subset Simulation for structural reliability problems are given in [16, 239, 238, 190].

Let the threshold event of interest be given by T, where T is defined as the region in the output distribution such that $\Psi = \chi_T = 1$ as in equation (4.3). To perform Subset Simulation, it is necessary to be able to define the event T in terms of a series of more likely events. Specifically, adopting the definition in [15], denote the monotone sequence of subsets converging to T as:

$$T_1 \subset T_2 \subset \cdots \subset T_m = T \tag{4.6}$$

Expressed in terms of set intersections, $T = T_m = \bigcap_{i=1}^{m} T_i$ for $i = 1, \ldots, m$.

The final subset T is denoted by T_m to make the index notation consistent. As a useful example of a selection of subsets, consider the probability that the maximum deformation, v, of a structural system exceeds some threshold value W. Then a decreasing subset series of failure events could be defined as $T_i = \{v > W_i\}$ where $W_1 < \cdots < W_m = W$.

Conditional probabilities of the subsets can be used to estimate the probability of T occurring, which is given by $P(T) = \overline{\Psi}$ where $\overline{\Psi}$ is defined as in equation (4.2). The subset probability derivation is simple calculation. Essentially, by basic probability theory definitions, $P(T_{i+1}|T_i)P(T_i) = P(T_i|T_{i+1})P(T_{i+1})$. But, as T is a monotone sequence, $P(T_i|T_{i+1}) = 1$ so $P(T_{i+1}|T_i)P(T_i) = P(T_{i+1})$. An induction argument, given in [15, 16], yields:

$$P(T) = P(T_1) \prod_{i=1}^{m-1} P(T_{i+1}|T_i) \tag{4.7}$$

Equation (4.7) states the probability $P(T)$ can be calculated by the product of $P(T_1)$ and each $P(T_{i+1}|T_i)$ for $i = 1, \cdots, m-1$.

If $P(T)$ is vanishingly small direct MCS techniques will not be efficient in calculating the event probability, as described in Section 4.2.3. By careful selection of intermediate threshold levels (the T_i values), a series of intermediate analyses can be carried out, each of which evaluate the $P(T_{i+1}|T_i)$ efficiently. The initial probability $P(T_1)$ can be estimated effectively by a direct (or potentially quasi) Monte Carlo approach [15, 16]. T_1 can be selected such that convergence of $P(T_i)$ requires only a comparatively small number of simulations. Sampling from the conditional distributions, $P(T_{i+1}|T_i)$, can be carried out by combining Markov Chain Monte Carlo with an additional phase of accept/reject sampling [15, 16]. Techniques for Markov Chain Monte Carlo are described in Section 4.2.5. The Subset Simulation accept/reject phase is described in Section 4.2.6.

4.2.5 Markov Chain Monte Carlo for high dimensional random fields

4.2.5.1 Markov Chain Monte Carlo overview

Markov Chain Monte Carlo (MCMC) is not a single technique, but rather a family of techniques for sampling from complicated distributions. The basic Markov Chain procedure is to sample from the target distribution by a random walk, moving between points probabilistically. The probability to transition from one point to another is given by a factor dependent on the target distribution probability density. MCMC techniques are useful when it is difficult to sample from a distribution directly. The original Markov Chain Monte Carlo was the Metropolis-Hastings algorithm [262]. The history of this algorithm is discussed in [313]. The basic format of all MCMC techniques are very similar to the Metropolis-Hastings technique. In this Section, after briefly defining Markov Chains and some of their properties, three Markov Chain Monte Carlo sampling techniques are described. These are Metropolis-Hastings, Componentwise Metropolis-Hastings and Gibbs sampling. As the performance of these three sampling algorithms are compared in Section 4.3 of this Chapter, it is useful to briefly describe these methods here, with reference to the details in [144, 55, 266, 311].

Markov Chain Monte Carlo integration relies on the convergence to a stationary distribution implied by the Fundamental Theorem of Markov Chains (Theorem 6.2 in [271]). A transition function, $Q(\cdot)$, is defined such that the Fundamental Theorem of Markov Chains is satisfied. If $f(Y)$ is a function of the random variables Y, the expectation $\mathbb{E}[f(Y)]$ can be estimated, after N sampling steps, by:

$$\overline{f}_N = \frac{1}{N}\sum_{t=1}^{N} f(Y_t) \tag{4.8}$$

where the function $f(Y_t)$ is evaluated at a sampled value $y^t \in Y_t$. The random samples are sampled using the transition function so that the random sample y^t comes from the distribution $Q(y^{t-1})$ (see §1.3.2 of [144] for more detail).

For stationary Markov Chains, the Central Limit Theorem holds and the error of $\overline{f}_N$ can be taken to be normally distributed by $\overline{f}_N \approx \mathcal{N}\left(\mathbb{E}\left[f(Y)\right], \frac{\sigma^2}{N}\right)$ [55]. This is identical to the MCS case, except that the estimate for σ^2 is more

complex for Markov Chains.

The variance of the estimate of $\mathbb{E}[f(Y)]$ in equation (4.8) is given by:

$$\sigma^2 = \mathrm{Var}[f(Y_i)] + 2\sum_{k=1}^{\infty} \mathrm{Cov}[f(Y_i), f(Y_{i+k}))] \tag{4.9}$$

The presence of the covariance term makes the estimation of the variance of a Markov Chain Monte Carlo integral substantially more difficult than in the direct Monte Carlo Simulation case. This is because the covariance term in equation (4.9) cannot be estimated directly. A detailed discussion of the issues regarding variance estimation for MCMC is given in [140] and §1.8 to §1.10 of [55].

From Theorem 3.1 in [140], for increasing k, Γ_k is is strictly positive, decreasing and convex. Then the chain variance can be estimated by summing the first m values of $\overline{\Gamma}_k$:

$$\sigma^2 = -\overline{\nu}_0 + \sum_{k=0}^{m} \overline{\Gamma}_k \tag{4.10}$$

where $\overline{\Gamma}_k = \overline{\nu}_{2k} + \overline{\nu}_{2k+1}$ and $\overline{\nu}_0 = \mathrm{Var}[f(Y_i)]$. The value of m is selected depending on the variance estimator used. The initial monotone sequence estimator selects m such that the $\overline{\Gamma}_k$ sequence are in the initial convex minorant. That is, m is increased until $\overline{\Gamma}_{m+1}$ breaks the convexity of $\overline{\Gamma}_0, \ldots, \overline{\Gamma}_m$. In this Chapter, the initial convex sequence estimator is used to compute all Markov Chain variances for the numerical analyses presented in Section 4.3.

The Markov Chain Monte Carlo algorithms applied to the numerical Subset Sampling problems in Section 4.3 are presented in the remainder of this Section.

4.2.5.2 The Metropolis-Hastings algorithm

Let Ω be a space from which random samples are to be drawn and $f(\omega)$ be a distribution proportional to the desired probability distribution $P(\omega)$. Given a symmetric proposal distribution $Q(\omega_i|\omega_j) = Q(\omega_j|\omega_i)$, the basic Metropolis-Hastings algorithm proceeds as follows:

1. Let the current sample be $\omega_i \in \Omega$

2. Generate a new potential sample, ω_p, from $Q(\omega_p|\omega_i)$.

3. Calculate the acceptance ratio:

$$\alpha = \frac{Q(\omega_i|\omega_p)f(\omega_p)}{Q(\omega_p|\omega_i)f(\omega_i)} = \frac{f(\omega_p)}{f(\omega_i)} \tag{4.11}$$

4. If $\alpha \geq 1$, then ω_p is more likely than ω_i and is accepted, so x_{i+1} is set to x_p.

5. If $\alpha < 1$, then the potential sample is less likely than ω_i. ω_p is accepted with probability α and $\omega_{i+1} = \omega_p$. If the sample is rejected, then the random walk does not transition to a new location and $x_{i+1} = x_i$.

6. Repeat.

The convergence of Metropolis-Hastings to the stationary distribution π follows from the discussion in Section 4.2.5.1. The choice of a symmetric proposal distribution coupled with the acceptance ratio technique guarantees that the detailed balance condition is satisfied. That this is the case is discussed in detailed in [72]. Then, as long as sampling is performed over an aperiodic and irreducible space, ergodic convergence to the stationary distribution follows [144].

4.2.5.3 Gibbs and Componentwise Metropolis-Hastings sampling for high dimensional spaces

In high dimensional spaces, Gibbs sampling is often used instead of Metropolis-Hastings [55]. The basic Gibbs sampling procedure is to carry out the Markov Chain random walk by re-sampling from only a single probabilistic degree of freedom per Monte Carlo step. This method is detailed in [134]. Gibbs sampling has two potential advantages. First, as Gibbs sampling does not require the acceptance ratio to be calculated (§1 of [55]), there is no need to deal with the problems associated with tuning the acceptance ratio in high dimensional spaces. This problem is described in [290]. Second, any numerical stability issues associated with estimating joint distribution sample probabilities in high dimensional spaces can be bypassed. However, componentwise sampling may exhibit slower convergence than methods which resample the entire sample vector every iteration [55].

For a random field with M degrees of freedom, calculating the total sample probability directly requires that the probability from each of the M marginal distributions are multiplied together to find the probability from the joint distribution $P(\omega \in \Omega)$. For a random field simulator that draws samples from independent marginal distributions before applying some correlation transform (either by matrix or spectral methods [118, 351]), the joint probability can be estimated by:

$$P(\omega \in \Omega) = \prod_{i=1}^{M} P(\omega_i) \tag{4.12}$$

Unfortunately, equation (4.12) is highly numerically unstable for even a small number of degrees of freedom. This problem can be resolved by simply taking log-probabilities, as described briefly in Section 4.2.5.4.

To avoid calculating acceptance ratios in high dimensional spaces, rather than using Gibbs sampling, many authors adopt Componentwise Metropolis-Hastings sampling. This method proceeds by performing a Metropolis-Hastings step for each independent degree of freedom in turn, as in [15, 238, 239]. This bypasses both the numerical stability issue above and the problems that can be faced when tuning the acceptance ratio as in equation (4.11) for high dimensional probabilistic spaces. Componentwise Metropolis-Hastings still, however, requires a much larger number of iterations than direct Metropolis-Hastings or other more advanced MCMC methods [55].

The problems solved by Gibbs and CMH sampling, primarily avoiding high dimensional acceptance ratio calculations, are not worth the additional cost incurred in the slow convergence of these methods for nonlinear FEM problems, as demonstrated empirically in Section 4.3. For RFEM, each iteration requires the evaluation of finite element equations. For nonlinear problems, these FEM evaluations are computationally very demanding and so the goal of efficient RFEM must be to minimise the number of FEM solutions. The Metropolis-Hastings methodology, adopted for the test case analyses presented in Section 4.3, is discussed in Section 4.2.5.5.

4.2.5.4 Log probabilities for avoiding numerical stability issues in Metropolis-Hastings

Avoiding Gibbs or Componentwise Metropolis-Hastings sampling requires resolving the numerical stability issue described in Section 4.2.5.3. This Section very briefly describes the log-probability workaround to this problem. Taking the logarithm of equation (4.12), the joint probability of a particular sample from the random field becomes:

$$\log\left(P(\omega \in \Omega)\right) = \log\left(\prod_{i=1}^{M} P(\omega_i)\right) = \sum_{i=1}^{M} \log\left(P(\omega_i)\right) \tag{4.13}$$

The MCMC acceptance ratio becomes:

$$\log \alpha = \frac{\sum_{i=1}^{M} \log\left(P(\omega_p)\right)}{\sum_{i=1}^{M} \log\left(P(\omega_i)\right)} = \sum_{i=1}^{M} \log\left(P(\omega_p)\right) - \sum_{i=1}^{M} \log\left(P(\omega_i)\right) \tag{4.14}$$

This simple modification allows for acceptance ratios to be calculated in a numerically stable fashion as the difference of logarithms.

4.2.5.5 Metropolis-Hastings and random fields for high dimensional MCMC

As stated in Section 4.2.5.3, Gibbs and Componentwise Metropolis-Hastings sampling typically converge more slowly than Metropolis-Hastings or other MCMC methods that perform a random walk by adjusting more than a single probabilistic degree of freedom per iteration when sampling is computationally expensive. Although, for example, Metropolis-Hastings may involve no transition from one sample point to another during a random walk iteration, this is computationally not a problem for combined Subset Simulation - RFEM type analyses. If the random walk remains in place, then there is no need to re-evaluate the FEM equations. Considering computational cost only, as long as the time taken to find a new random sample is less than the time taken to evaluate the FEM equations, Metropolis-Hastings will outperform Gibbs sampling for an entire Subset Simulation - RFEM computation. In Section 4.3, a numerical comparison of Componentwise Metropolis-Hastings sampling and Metropolis-Hastings sampling is presented.

This Section details a method for Metropolis-Hastings sampling from a random field discretised as per the discussion in Section 4.2.2 suitable for use in Subset Simulation - RFEM analyses. The random walk procedure adopted for the analyses in Section 4.3 follows the algorithm in Section 4.2.5.1. Acceptance probabilities are calculated using the logarithmic formula in Section 4.2.5.4. The transition function is given by a multivariate Gaussian distribution. Efficient search through the probabilistic space is performed, with reference to §6 of [144], by altering the M dimensional γ vector in equation (4.4). That is, the random stepping is performed on the space of uncorrelated Gaussian samples given by the vectors γ.

In more detail, let the random field sample at the current stage of the Metropolis-Hastings sampling be given by $\beta^t = G\gamma^t$. To calculate β^{t+1}, each of the entries in γ^{t+1} can be updated before applying G. Specifically, set γ^{t+1} equal to a random sample from $\mathcal{N}(\gamma^t, \sigma^2 I)$ where I is the M dimensional identity matrix and σ is a scalar parameter that must be tuned to find an acceptable convergence rate for the MCMC stages. Then each entry γ_i^{t+1} is a random sample from $\mathcal{N}(\gamma_i^t, \sigma^2)$. Finally, given the updated set of uncorrelated Gaussian γ^{t+1} samples, β^{t+1} is given by:

$$\beta^{t+1} = G\gamma^{t+1} \text{ where } \gamma_i^{t+1} \sim \mathcal{N}(\gamma_i, \sigma^2) \; \forall i \in M \tag{4.15}$$

Such a sampling scheme is obviously irreducible in that the Gaussian transition function has a non-zero probability to jump from any state to any other state in a single time step, so that $P(i|j) > 0 \; \forall i, j$ where i, j are possible configurations of γ. Further, because of this, the sampling methodology is also aperiodic. To see this, consider that each state can return to itself in a minimum of two steps by jumping from one state and then immediately jumping back, i.e. $i \to j \to i$. Each state can return in three steps by jumping via an intermediate state k back to itself: i.e. $i \to j \to k \to i$. As the greatest common divisor of 2 and 3 is 1, the Markov Chain is aperiodic. Finally, the acceptance ratio technique of the Metropolis-Hastings method is constructed to ensure that the detailed balance condition is satisfied and so that the stationary distribution exists as discussed in Section 4.2.5.2. Then, by the Fundamental Theorem of Markov Chains discussed in Section 4.2.5.1, the time average of the sampling methodology will converge to the average over all possible samples for a large enough number of samples regardless of the initial sample.

By adjusting the γ vector, rather than adjusting β, the convergence rate of the integral should improve as the correlation structure of the random field does not prevent the random walk from finding high probability portions of the search space (see §6 and in particular Figure 6.1 of [144]).

4.2.6 Subset Simulation accept-reject sampling

To compute the conditional probability in equation (4.7), an additional accept/reject stage can be added onto the MCMC algorithm. This method is described in [15]. Modifications to this technique are discussed in [290]. Consider computing the probability $P(T_{i+1}|T_i)$. After generating a proposal sample, the sample is required to satisfy $\Psi_i(u_a(\omega)) = 1$. That is, if the proposed sample causes the solved equations to reach the minimal subset threshold level, the sample can be accepted. Otherwise, the sample is rejected and ω is left unchanged. The probability of $P(T_{i+1}|T_i)$ can then be estimated by:

$$P(T_{i+1}|T_i) = \frac{1}{K}\sum_{j=1}^{K}\Psi_{i+1}(u_a(\omega_j)) \tag{4.16}$$

where K is the number of accepted samples. Acceptance or rejection of a sample (subset accept/rejection) at this stage is distinct from the MCMC sampling accept/reject stage that is required for both Metropolis-Hastings and Componentwise Metropolis-Hastings sampling.

4.2.7 Subset Simulation error estimation

In Section 4.2.3, a well known technique for estimating the standard error for MCS was presented. The estimation of errors for Subset Simulation is significantly more complicated. The estimation of the standard deviation of estimated Subset Simulation responses have been presented by others (see §5.3 [16]). The calculation of confidence intervals is, however, more difficult. The derivation of confidence interval estimates are presented in this Section.

The estimate of $P(T)$ is, with reference to equation (4.7), made by taking the product of several estimated subset probabilities. From the discussion in Sections 4.2.3 and 4.2.5.1, each of the subset probabilities $P(T_1)$ and $P(T_{i+1}|T_i)$ $\forall i$ are normally distributed by the Central Limit Theorem. Then estimated value

of $P(T)$ is distributed according to the product of several normal distributions:

$$P(T) \approx \prod_{i=1} \mathcal{N}(\mu_i, \sigma_i^2); \tag{4.17}$$

Let Z_k be the product distribution formed from random variables X_i so that $Z_k = \prod_{i=1}^{k} X_i$. From [152], the expectation and variance of the product of N independent probability distributions are given by:

$$\mathbb{E}[Z_k] = \prod_{i=1}^{k} \mathbb{E}[X_i]$$
$$\mathrm{Var}(Z_k) = \mathrm{Var}(X_k)Var(Z_{k-1}) + \mathrm{Var}(X_k)\mathbb{E}\left[Z_{k-1}\right]^2 + \mathrm{Var}(Z_{k-1})\mathbb{E}\left[X_k\right]^2$$

The product of normal distributions is not a normal distribution. Then, to estimate confidence intervals for $P(T)$ it is necessary to calculate the confidence interval limits for the product distribution.

From §4.7, Theorem 3 and Corollary 4 from [161] and Theorem 3, §4.4 in [315], the density, $f^Z(z)$, for the product $Z = XY$ of two random variables with probability density $f^X(x)$ and $f^Y(y)$ is given by:

$$f^Z(z) = \int_{-\infty}^{\infty} f^Y\left(\frac{z}{x}\right) f^X(x) \frac{1}{|x|} dx \tag{4.18}$$

Next, consider the case of the product of k distributions. Let $Z_1 = X_1$ and $Z_k = \prod_{i=2}^{k} X_i$. Let the density function for any Z_i and $X_i(x_i)$ be given by $f_i^Z(z_i)$ and $f_i^X(x_i)$ respectively. Then for $i = 1$, $f_1^Z(z_1) = f_1^X(x_1)$. It is shown in the Chapter Appendix by means of an induction argument that for $(k \geq 2) \in \mathbb{N}$:

$$f_k^Z(z_k) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f_k^X\left(\frac{z_k}{w_{k-1}}\right) \left[\prod_{i=2}^{k-1} f_i^X\left(\frac{w_i}{w_{i-1}}\right) \frac{1}{|w_{i-1}|}\right]$$
$$\times \frac{1}{|w_{k-1}|} f_1^X(w_1) dw_1 \ldots dw_{k-1} \tag{4.19}$$

Following from [224] Definition 5.1, the limits for a confidence interval of $1 - \alpha$, over the product distribution can be estimated by finding $\left[z^L, z^U\right]$

such that:

$$1 - \alpha = \int_{z^L}^{z^U} f^Z(w) dw \tag{4.20}$$

The product density function is, in general, asymmetrical. There are multiple methods to express confidence intervals for an asymmetrical probability distribution. One technique, which will be adopted in this Chapter, is to calculate confidence intervals with equal probability mass above and below the median. Let the median of Z_k be denoted $\tilde{Z}_k$. Then the upper and lower confidence intervals for Subset Simulation can be found by calculating:

$$z_k^L \text{ such that } \int_{z_k^L}^{\tilde{Z}_k} f_k^Z(w_k) dw_k = \frac{1-\alpha}{2}$$

$$z_k^U \text{ such that } \int_{\tilde{Z}_k}^{z_k^U} f_k^Z(w_k) dw_k = \frac{1-\alpha}{2}$$

where $f_k^Z(z_k)$ is given by equation (4.19). Binary search (see [80]) can be used to rapidly locate the confidence interval limits and the median. For small k, the integral problems above can be efficiently estimated by quadrature. For $k > 3$ or 4, high dimensional integration techniques such as MCS can be used.

4.2.8 Summary and discussion

The extension of the Random Finite Element Method to include Subset Simulation, based on [15, 329, 239], was described. A method for applying Metropolis-Hastings, rather than Gibbs or Componentwise Metropolis-Hastings, sampling for probabilistic FEM using Subset Simulation when random fields are present was described. The method presented is not dependent on the particular random field simulation methodology used. Any of the common field generators, such as those in [118, 43] or any other reasonable method, could be used. Further, the extension of Subset Simulation for FEM techniques to full Metropolis-Hastings sampling should allow for the more advanced MCMC techniques that rely on adjusting all probabilistic degrees of freedom simultaneously to be applied in the future. For example, Hamiltonian Monte Carlo and other techniques detailed in [55] may be an interesting avenue for future work. Further, the derivation of the full product density function will allow for more detailed analysis of Subset Simulation convergence than can be achieved

by simply evaluating the product distribution variance.

4.3 Applications

4.3.1 Introduction

The theoretical developments in Section 4.2 are tested empirically in this Section. Two probabilistic numerical analyses of a footing load on soil are presented. The general reliability analysis problem described by equation (4.2) is explored by finite element stress-strain analysis.

The first analysis compares the performance of Subset Simulation to Monte Carlo Simulation (MCS) and SSFEM results published in §4 of [359] and [360] on the reliability analysis of a footing on linear-elastic soil. This first set of analyses will be referred to as the verification study and will serve to demonstrate that the MCS and MCMC implementations give similar results to independent results by SSFEM. Further, because the verification study analyses consist of a series of deterministic linear elastic FEM problems, a very large number of analyses can be completed in a reasonable amount of time. In contrast, nonlinear FEM analysis is far more computationally demanding. By carrying out a large number of linear simulations, the convergence characteristics of the MCS and MCMC techniques presented can be compared to a higher degree of accuracy than can be achieved with plastic analyses. The results of the verification study analyses are presented in Section 4.3.3.

The second analysis modifies the problem presented in [360] by altering the constitutive model to an elastoplastic Mohr-Coulomb model. The mean and standard deviation of the modified material parameters were selected to be like the typical values encountered in practice. Further, the parameters were chosen such that the probability of failure using MCS was very small allowing for the relative performance of Subset Simulation and MCS to be compared. The second analysis will be referred to as the comparative study. The results for the comparative study are presented in Section 4.3.4.

4.3.2 Common problem description - verification and comparative study

4.3.2.1 Problem geometry, random fields and FEM details

The footing problem geometry is a two dimensional rectangle of soil 30 m high by 120 m wide with a centrally placed load. To match the values in [360], a deterministic load of $q = 200$ kPa was applied at the top of the soil over a width of 10 m for all analyses. The same mesh as that presented in [360] with 80 elements is used for all analyses. The problem geometry with the finite element mesh overlaid is shown in Figure 4.1. As in [360], a plane strain model is used for all analyses.

Technically, including random material property parameters disrupts the reflective symmetry of the problem. However, for comparison with [360], only half of the problem domain was meshed for the analyses in this Chapter. This does also raise the question as to whether two-dimensional modelling is appropriate for this scenario. The plane strain analysis effectively assumes perfect infinite 'into-the-page' correlation of the spatially random material properties. For small vertical deformations of the footing (compared to the scale of fluctuation of the material property parameters), such an assumption is unlikely to cause large model errors. Small approximation errors regarding the physical properties of soils can be considered to be negligible for the purposes of comparing different probabilistic reliability estimation techniques.

Convergence tolerance for all finite element analyses was set at 1×10^{-6}. A maximum of 5000 Newton-Raphson iterations was adopted (mainly relevant for the nonlinear comparative study analyses). The y-displacements were fixed to 0 along the base of the problem domain and the x-displacements were fixed to 0 along the left and right edges. With these parameters, the maximum displacement $u = 54.2$ mm occurs at the centre of the loaded area, matching the result in [360].

All random fields are modelled as independent from one another in all analyses. All random fields were simulated using the same correlation structure. Specifically, the distribution of material property inputs was taken to be a multivariate Gaussian, $\mathcal{N}(\mu, \Sigma)$, with an exponentially decaying autocorrelation

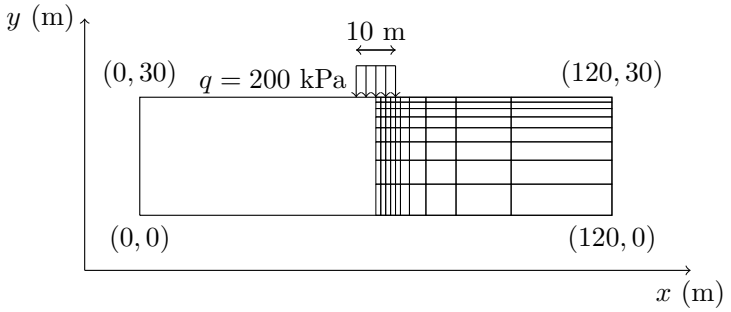


Figure 4.1: Footing problem geometry and mesh for both the verification and comparative studies.

function, from [118]:

$$C(\tau_x(p_1, p_2), \tau_y(p_1, p_2)) = \exp\left(-\frac{\tau_x}{\theta_x} - \frac{\tau_y}{\theta_y}\right) \tag{4.21}$$

where p_1 and p_2 are the (x, y) coordinates of two points within the random field. $\tau_x(p_1, p_2) = |p_{2x} - p_{1x}|$, $\tau_y(p_1, p_2) = |p_{2y} - p_{1y}|$, that is, τ_x and τ_y are the l_1-norm distance between points along each axis. In this Chapter, the variance of a random parameter is expressed in terms of coefficient of variation, δ, which is the ratio of the standard deviation σ and the mean μ so $\delta = \frac{\sigma}{\mu}$.

The material property parameters and coefficients of variation were taken from the typical ranges for clays listed in Table 1 of [297]. The parameters were selected to ensure that the calculated probability of failure would be small enough that a Subset Simulation, rather than direct MCS, would be required to estimate the system reliability. A slight dilation angle, 0.1°, was adopted for all analyses.

For the probabilistic finite element analysis, each discrete simulation was generated by sampling values, ω, from the appropriate random fields and then constructing the stiffness matrix, $\mathbf{K}(\omega)$. After assembling $\mathbf{K}(\omega)$, the finite element equations, $\mathbf{K}(\omega)u(\omega) = f$ are solved for $u(\omega) = \mathbf{K}^{-1}(\omega)f(\omega)$.

A parallel implementation of the eigenvalue random field simulator described in [158, 118] was used. In particular, [158] details a method to accurately simulate Gaussian random fields with positive definite correlation functions even in the presence of numerically singular covariance matrices by identifying linear dependencies in the random field correlation structure. As the correlation function presented in equation (4.21) is trivially positive definite and the ran-

dom field generator described in [158] was used, singular and non-positive definite covariance matrices were not an issue for the analyses presented in this Chapter.

Numerical issues resulting from the use of Gaussian distributions for non-negative material property parameters were handled by truncating the distributions. If the numerical value sampled were such that the numerical problems were unstable and could not be solved, these results were discarded and the Markov Chain (if being used) did not accept the next step. For the parameters tested, the probability to sample a value causing a truncation was several orders of magnitude below the probability of interest and as such the error caused by such truncation was much smaller than the confidence intervals bounding the mean response. This situation could be improved by the application of non-Gaussian random field models.

All code was written in C++ using freely available software libraries. Parallel random field simulation code was written using PETSc [26, 25, 24] (nonlinear matrix solvers, in particular Jacobian-Free Newton-Krylov methods) and Elemental [303] (dense linear algebra). The finite element simulation, including the plasticity model, code is a part of the MOOSE framework [131] which also makes use of PETSc.

For the elastic analyses, the elastic strain energy should remain positive definite at the scale of elements within the mesh. This is because the Young's Modulus and Poisson's Ratio are restricted to be positive in all simulated cases, forcing positivity of the element stiffness matrices. This should hold in all cases (up to machine precision and assuming no errors in the software used). Further, for elastoplastic analysis, the plastic energy dissipation is required to remain positive definite in the plastic domain. The software used for all analyses, the MOOSE framework [131], employs a return mapping scheme designed to ensure the physically correct behaviour of strain hardening materials (as used in the numerical analyses presented in this Chapter). The return mapping scheme is described in detail in the MOOSE framework documentation in [133].

All Metropolis-Hastings and Componentwise Metropolis-Hastings sampling adopted a Gaussian transition probability distribution with a standard deviation $\sigma = 0.25$ in the space of uncorrelated, standard normal samples. These samples form the entries of γ in equation (4.4).

4.3.2.2 Reliability assessment limit state function

Both studies assess the probability $P(u(\omega) > u_0) = P(T)$ where ω represents the random dimension of the problem, $u(\omega)$ is the maximum displacement at the centre of footing area and u_0 is a limit state deformation of interest. As in equation (4.3), the limit state reliability can be represented as an indicator function:

$$\Psi(u(\omega)) = \chi_T = \begin{cases} 1 \text{ if } u(\omega) > u_0 \\ 0 \text{ otherwise} \end{cases} \tag{4.22}$$

For comparison with the results published in [360], the following limit state deformations were used to calculate $P(u(\omega) > u_0)$ for all analyses: $u_0^1 = 60$ mm, $u_0^2 = 80$ mm, $u_0^3 = 100$ mm, $u_0^4 = 120$ mm, $u_0^5 = 150$ mm. For Subset Simulation in particular, the notation $P(T_i) = P(u(\omega) > u_0^i)$ will also be used. Additionally, from [360], the expected $P(u(\omega) > u_0)$ should decrease by approximately one order of magnitude for each increase in the limit state function value, i.e. $P(T_{i+1}) \approx 10^{-1} \times P(T_i.)$ From this point on in this Chapter, the notation $P(u(\omega)) > u_0)$ will be written $P(u > u_0)$.

4.3.3 Verification study - linear elastic footing

4.3.3.1 Problem description

For the verification study, the performance of Subset Simulation and MCS was compared to SSFEM results published in [360]. A linear elastic material model was adopted with the properties given in Table 4.1. Note that the reliability analysis results presented in [360] are expressed as reliability indices, β. In this Chapter the probability for the displacement, u, to exceed the limit state displacement u_0 will be preferred and so the results from [360] are presented here after conversion to probability, $P(u > u_0)$. Both β and $P(u > u_0)$ are simple transforms of each other. The conversion is $P(u > u_0) = 1 - \Phi(\beta)$ where Φ is the cumulative standard normal distribution, see §13.6 in [21] for detailed discussion.

The following analyses were conducted for the verification study. First, a comparison of SSFEM and MCS was conducted. A large number (1×10^6) of FEM simulations were carried out. The results of these analyses were used to

estimate $P(u > u_0)$ for each $u_0 = \{60, 80, 100, 120, 150\}$ mm. These were then compared to the range of results presented in [360] to check that there was a reasonable agreement. It is worth noting that [360] argues that the accuracy of the SSFEM analysis is expected to decrease for increasing u_0. From the discussion in Section 4.2.3, it was also anticipated that a larger number of FEM analyses will be required for MCS to accurately estimate reliabilities for larger u_0, that is, further from mean output responses.

The verification study next compared the performance of direct MCS with Subset Simulation. Three MCMC samplers were tested: Metropolis-Hastings (MH), Componentwise Metropolis-Hastings (CMH) and Gibbs sampling. These three samplers were detailed in Section 4.2.5. The performance of MCS and Subset Simulation was evaluated by estimating the number of runs required to reach a particular relative accuracy of the estimated mean value for each $u_0 = \{60, 80, 100, 120, 150\}$ mm.

Parameter	Symbol (Units)	Mean, μ	Coeff. of Variation, δ
Applied load	q (kPa)	200	NA
Young's Modulus	E (kPa)	50	0.2
Poisson's Ratio	ν (NA)	0.2	NA
Correlation length (x)	θ_x (m)	∞	NA
Correlation length (y)	θ_y (m)	30	NA

Table 4.1: Parameters used for the Verification Study. Note: NA = Not applicable. These parameters match those published in [360].

4.3.3.2 Numerical results

Monte Carlo Simulation was used to establish base line results for the verification study. A single MCS analysis of one million FEM simulations was conducted. Trace plots showing the convergence of the MCS simulation are presented in Figure 4.3. The numerical values of the MCS simulation values for $P(u > u_0)$ for $N = 1 \times 10^5$ and $N = 1 \times 10^6$ FEM simulations are listed in Table 4.2. The accuracy of the estimates in Table 4.2 are given as relative errors on the 95% confidence intervals. The confidence intervals were calculated by the methods described in Section 4.2.3. The relatives errors for MCS

are expressed in terms of the 95% confidence intervals as percentages:

$$\text{Rel. Err. } (\%) = \frac{1.96 \times \frac{\sigma_N}{\sqrt{N}}}{P(u > u_0)} \times 100 \tag{4.23}$$

u_0 (mm)	MCS - 1×10^5 FEM runs $P(u > u_0)$ ($\pm$ Rel Err.)	MCS - 1×10^6 FEM runs $P(u > u_0)$($\pm$ Rel Err.)
60	2.79×10^{-1} ($\pm$ 9.97×10^{-1} %)	2.78×10^{-1} ($\pm$ 3.16×10^{-1} %)
80	1.40×10^{-2} ($\pm$ $5.19\times$ 10^{0} %)	1.44×10^{-2} ($\pm$ 1.62×10^{0} %)
100	7.90×10^{-4} ($\pm$ 2.20×10^{1} %)	9.09×10^{-4} ($\pm$ $6.50\times$ 10^{0} %)
120	4.00×10^{-5} ($\pm$ 9.80×10^{1} %)	1.21×10^{-4} ($\pm$ 1.78×10^{1} %)
150	1.00×10^{-5} ($\pm$ 1.96×10^{2} %)	1.20×10^{-5} ($\pm$ 5.66×10^{1} %)

Table 4.2: Summary of Monte Carlo Simulation results (1×10^5 and 1×10^6 FEM simulations) for linear elastic verification study. Probabilities $P(u > u_0)$ are expressed as factors between 0 and 1. Relatives errors are calculated for 95% confidence intervals, see equation (4.23).

Figure 4.2 compares the MCS analysis for both 1×10^5 and 1×10^6 simulations in Table 4.2 to SSFEM results published in Table 4 of [360]. Note that multiple values are provided for each u_0 in [360]. Each of these published values has been marked with a triangle on the plot. The MCS values are shown as 95% confidence interval bands. Figure 4.2 also displays the 95% confidence interval bands for Subset Simulation by MH, CMH and Gibbs sampling using 1×10^5 FEM simulations at each subset level. The confidence intervals for Subset Simulation were calculated by the method detailed in Section 4.2.7.

The convergence versus number of FEM simulations is shown in Figure 4.3, which demonstrates the estimated mean and 95% confidence intervals for MCS, MH, CMH and Gibbs sampling for each u_0. Note that in Figure 4.3, the x-axis labels are the number of FEM solutions, N, but the axis scale is $\frac{1}{\sqrt{N}}$ as this scale best demonstrates the convergence rate of MCS. The Subset Simulation results in Figure 4.3 are shown for a large number of runs to demonstrate that the estimates converge for increasing N. Additionally, to compare to the convergence of MCS, note that the number of runs for Subset Simulation is expressed as a cumulative total. That is, the accuracy for Subset Simulation is estimated assuming that the same number of runs was used for each earlier level. For example, an estimate of $P(u > u_0^3)$ with $N = 3 \times 10^3$ is calculated assuming 1×10^3 runs were used for each $P(u > u_0^1)$, $P(u > u_0^2)$ and $P(u > u_0^3)$. In practice, a Subset Simulation Markov Chain analysis would

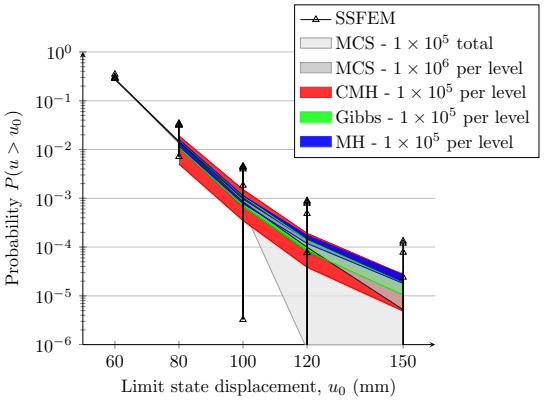
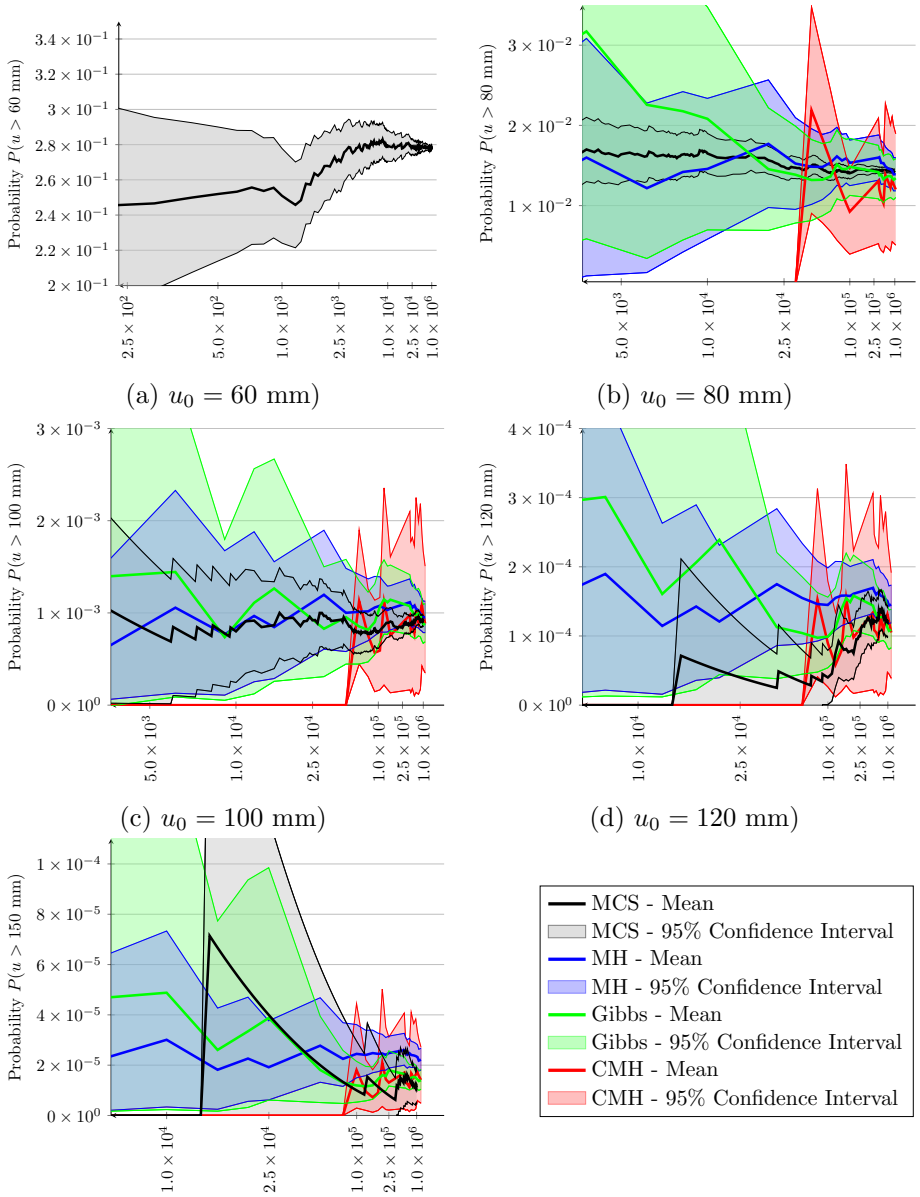


Figure 4.2: Verification study - SSFEM, MCS and Subset Simulation reliability analysis comparison. All probabilities expressed as a factor between 0 and 1. SSFEM results are from the results published in Table 4 [360], each published value is drawn as a single triangle. MCS and Subset Simulation values are shown for 95% confidence intervals. MCS values are shown for 1×10^5 and 1×10^6 total FEM simulations. Subset Simulation values are shown for 1×10^5 FEM simulations at each level u_0.

more likely run until a fixed tolerance or number of simulations was met. This approach is used for the comparative study. For the verification study, however, a large number of simulations was used to demonstrate the convergence of the sampling methodologies.



(a) $u_0 = 60$ mm)

(b) $u_0 = 80$ mm)

(c) $u_0 = 100$ mm)

(d) $u_0 = 120$ mm)

(e) $u_0 = 150$ mm)

(f) Legend: All plots

Figure 4.3: Verification Study - MCS vs Subset Simulation convergence. The x-axis for all plots shows labels for N = number of FEM simulations with scale $\frac{1}{\sqrt{N}}$. Shaded areas indicate 95% confidence interval regions for each sampling methodology (MCS, MH, CMH, Gibbs). The number of FEM simulations for Subset Simulation is calculated as a cumulative total for each $T_{i+1}|T_i$, including N for initial MCS step, T_1.

4.3.3.3 Discussion

The results of the verification study indicate that SSFEM, MCS and Subset Simulation all converged to similar estimates $P(u > u_0)$ for each u_0. MCS and SSFEM both perform well for closer to mean responses, e.g. for $u_0 = 60$ and 80 mm. For further from mean responses, the performance of both MCS and SSFEM degrade. The poor convergence of SSFEM for far from mean responses is to be expected based on the discussion in [360]. The worsening performance of MCS far from the mean is also expected, based on the discussion in Section 4.2.3 of this Chapter.

With reference to Figures 4.2 and 4.3, for $u_0 = 80$ mm MCS clearly outperforms Subset Simulation. However, for $u_0 = 100$ mm, Subset Simulation begins to outperform MCS. For $u_0 = 120$ and 150 mm, Subset Simulation performs significantly better than MCS. These empirical results support the anticipated response that Subset Simulation should converge more rapidly than MCS for far from mean responses. It is noted that for the linear elastic analyses, there is no significant variation in the run time to solve each FEM simulation and so the total number of simulations is an adequate measure of the performance.

For Subset Simulation, the choice of sampling methodology was observed to be significant. MH was shown to converge the most rapidly and exhibited the least oscillation about the estimated response. Gibbs sampling displayed the second best performance, with similar but slightly worse convergence than MH. Although CMH did eventually converge to the same estimated value as MH and Gibbs sampling for each u_0, the performance was significantly worse. The fluctuations about the estimated value were significantly higher. Further, presumably because the CMH chains mixed poorly, when estimating $P(T_{i+1}|T_i)$, a large number of simulations had to be conducted. This is in contrast to the more rapid convergence displayed by both MH and Gibbs sampling.

Overall, it was observed that SSFEM, MCS and Subset Simulation converged to similar values for the estimated probability to exceed the limit state threshold. For further from mean responses, Subset Simulation converged more rapidly than MCS. MH was found to be the most efficient MCMC methodology, followed by Gibbs and then CMH sampling.

4.3.4 Comparative study - elastoplastic soil

4.3.4.1 Problem description

The comparative study modifies and extends the problem from the verification study to include an elastoplastic constitutive model describing the onset of yield within the soil. The elastic parameters, Young's Modulus and Poisson's ratio are modelled using random fields. In addition, the parameters for the yield state are also modelled as random fields within the soil. The use of multiple random fields in this manner is not a particular challenge for a sampling-type methodology (e.g. MCS and MCMC) assuming that a deterministic solution can be evaluated for each sample. The particular material parameters used for the analyses presented were selected to ensure that the probability of failure as calculated by MCS would be very low in order to test the relative performance of Subset Sampling. Further, the material parameters were selected to be within the ranges typical of a realistic soil.

The elastoplasticity model used is a cap-smoothed variant of the Mohr-Coulomb model, detailed in [2]. Although this model is the simplest of those commonly used in the engineering of soils (see [272]), it is sufficiently complicated for testing the performance of MCS and Subset Simulation. Further, the techniques presented in this Chapter could be extended to other models. From §7 in [272], the basic Mohr-Coulomb model says that if the shear stress, τ, reaches a critical value on any plane within the material then yield will occur. The yield condition is given by:

$$\tau = \sigma \tan(\phi) + c \tag{4.24}$$

where σ is the normal stress to the given plane, ϕ is the internal friction angle and c is the material cohesion. The parameters ϕ and c define the onset of yield within the material. For numerical stability, the analyses presented also include a slight dilation angle ψ which causes the material to harden while yielding, see §8 of [272]. The plasticity model used in the analyses is described in the MOOSE framework software [131] documentation detail in [132]. In particular, it is noted that the Mohr-Coulomb framework adopted uses a non-associated flow rule if the dilation angle is not equal to the friction angle (as is the case for the analyses in this Chapter).

For the simulations, the random field variables are stored at the finite element mesh integration points. As FEM is an element-scale method, the pointwise positive definiteness guarantees that are required can only be made on the scale of the mesh elements. In effect, the choice of a mesh type discretisation is equivalent to filtering out of high frequency fluctuations of the random field. Such approximations are necessary when translating mathematical techniques to practical numerical algorithms.

The parameters used for the comparative study are summarised in Table 4.3. The correlation lengths chosen were selected based on the typical values in [194] and the fact that the critical correlation length tends to be on the dimensional scale of the problem domain [118].

The Mohr-Coulomb parameters were chosen to be similar to the values found for real clays. For example, [387, 298] find mean values of $c = 36$ kPa and $\phi = 25°$. The analyses in this Chapter have increased the cohesion value to 100 kPa to ensure that the probability of failure is small enough to test whether Subset Simulation can outperform MCS. Although 100 kPa is approximately three times larger than the values reported in [387, 298], it is equal to the value used in [116] and as such is within the region of typical values used for similar problems. The coefficients of variation adopted for c and ϕ are also typical for footing problems of the type analysed in this Chapter, see also [71, 116].

Parameter	Symbol (Units)	Mean, μ	Coeff. of Variation, δ
Applied load	q (kPa)	200	NA
Young's Modulus	E (kPa)	50	0.2
Poisson's Ratio	ν (NA)	0.2	NA
Correlation length (x)	θ_x (m)	30	NA
Correlation length (y)	θ_y (m)	10	NA
Cohesion	c (kPa)	100	0.2
Friction Angle	ϕ (Degs.)	25	0.2
Dilation Angle	ψ (Degs.)	1	NA

Table 4.3: Parameters used for the Comparative Study. Note: NA = Not applicable.

The comparative study assessed the relative performance of MCS and Subset Simulation by MH, CMH and Gibbs sampling for the nonlinear FEM problem described. The computational complexity of elastoplastic FEM problems is much higher than for linear elastic analyses. Then, for sampling-based prob-

ability estimation, the number of FEM simulations can be expected to be the time critical component of rare event simulation. The introduction of material yielding will induce multimodality in the output distribution not present in the verification study, decreasing the ability of the Markov Chains to mix easily when compared to the linear elastic case.

The aim of the study was to confirm whether Subset Simulation could converge more quickly than MCS for small probability of failure problems. The study also aimed to empirically (i.e. by observation) confirm the expected result that MH should converge more rapidly then CMH or Gibbs. This result was expected for two reasons. First, MH outperformed both MH and Gibbs for the verification study. Second, the more complex shape of the output distribution may prevent componentwise samplers from being able to efficiently explore the state space.

The comparative study was conducted to more closely match the way a real analysis might proceed. First, MCS was run to estimate $P(u > 60 \text{ mm})$ to a fixed relative error, see equation (4.23), of 1%. In the verification study, each subset level probability was estimated assuming each earlier level had used the same number of simulations. For the comparative study, each subset level MCMC estimate was made using the fixed accuracy estimate for $P(u(\omega) > 60 \text{ mm})$. Each MCMC analysis was run as a single chain for 1×10^5 FEM simulations.

To check the results, MCS was run for a total of 1×10^6 simulations. For analysis in practice, running a very large MCS analysis as well as Subset Simulation would defeat the purpose of using Subset Simulation at all. The MCS analysis with 1×10^6 FEM simulations was only intended to confirm that Subset Simulation did indeed converge to a reasonable result and indicate that Subset Simulation can therefore perform well in practice. For nonlinear FEM analysis, the time to solve each discrete FEM problem can vary significantly depending on the input parameters. By fixing the number of simulations allowed, the total run time of each of the Subset Simulation MCMC methodologies can be compared.

4.3.4.2 Numerical Results

The initial 1% fixed tolerance MCS analysis required 1.2×10^5 simulations. For 1.2×10^5 simulations, the probability to exceed $u_0 = 60$ mm was $P(T_1) = 0.257 \pm 0.96\%$. This value was used to estimate each subsequent $P(T_{i+1}|T_i)$ for Subset Simulation. All Subset Simulation results presented use this fixed value of $P(T_1)$ for 1.2×10^5 simulations for all estimated $P(T_i)$ for $i > 1$.

Figure 4.4 demonstrates the estimated probabilities for each u_0 after completing all FEM simulations. Probability estimates are expressed as 95% confidence interval regions. The results for Subset Simulation in Figure 4.4 were calculated using 1×10^5 FEM simulations for all Subset Levels. The number of runs per subset level were counted cumulatively. For example, the estimated values for $P(u > 100$ mm$)$ were found using a total of 3.2×10^6 FEM simulations. These 3.2×10^6 runs are the total of 1.2×10^6 MCS runs completed to estimate $P(u > 60$ mm$)$, followed by 1.0×10^5 MCMC runs for $P(u > 80$ mm$)$ and finally 1.0×10^5 MCMC runs for $P(u > 100$ mm$)$. Note that for the probability to exceed $u_0 = 150$ mm, random sampling by MCS failed to detect a single event in 1×10^6 iterations. Thus, the estimated probability $P(u > 150$ mm$)$ by MCS is less than 1 in 1×10^6.

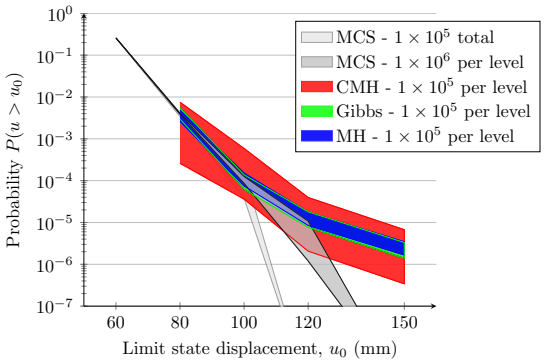
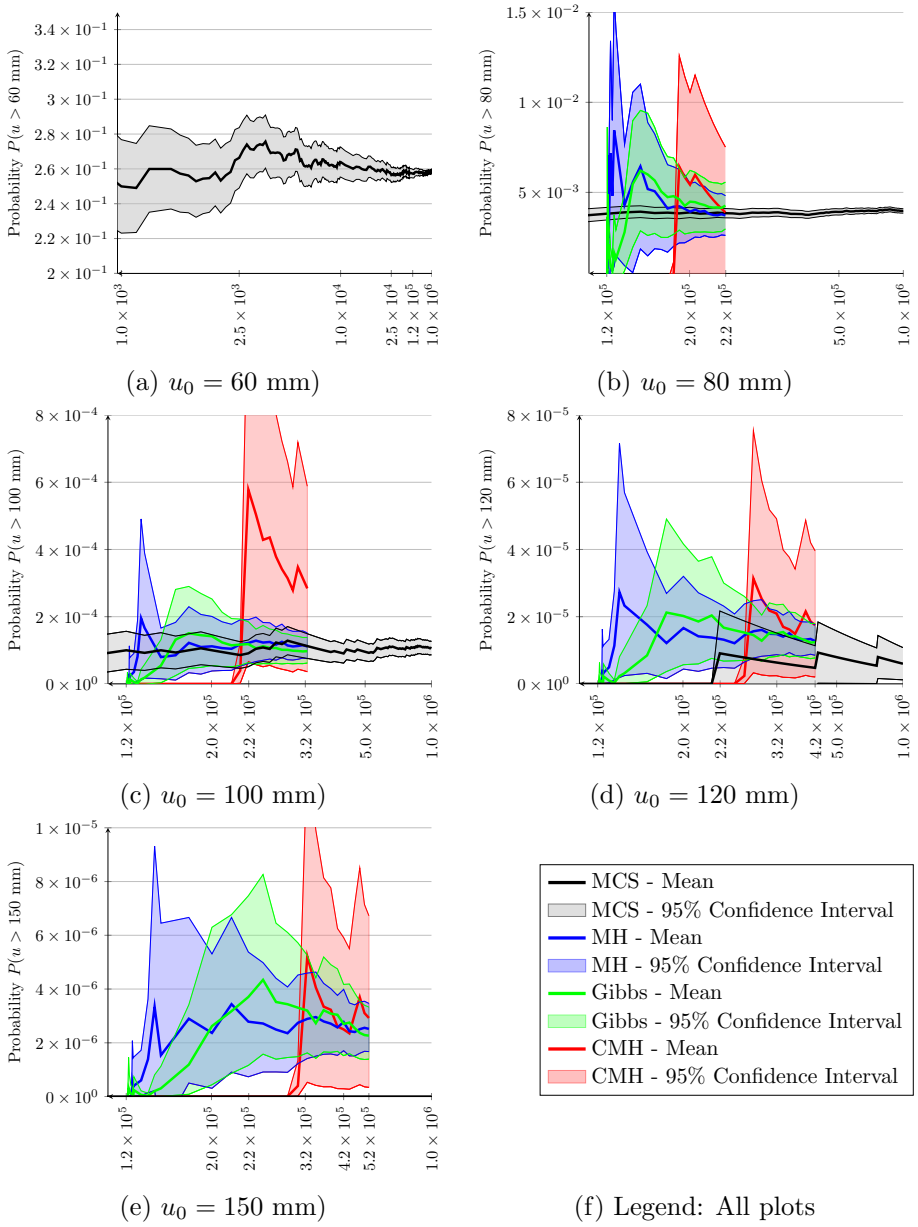


Figure 4.4: Comparative study - MCS and Subset Simulation reliability analysis convergence comparison. All probabilities expressed as a factor between 0 and 1. MCS and Subset Simulation values are shown for 95% confidence intervals. MCS values are shown for 1×10^5 and 1×10^6 total FEM simulations. Note that for 1×10^5, no displacements were detected greater than 120 mm. For 1×10^6 MCS simulations, no displacements were detected greater than 150 mm. Subset Simulation values are shown for fixed 1% relative error for $P(u > 60$ mm$)$ and 1×10^5 FEM simulations at each subsequent u_0 level.

Figure 4.5 shows trace plots of the mean and 95% confidence regions for each

u_0 as estimated by MCS, MH, CMH and Gibbs sampling. Note that, as in Figure 4.3, the x-axis labels are the number of FEM solutions, N, but the axis scale is $\frac{1}{\sqrt{N}}$ as this scale best demonstrates the convergence rate of MCS.



(a) $u_0 = 60$ mm)

(b) $u_0 = 80$ mm)

(c) $u_0 = 100$ mm)

(d) $u_0 = 120$ mm)

(e) $u_0 = 150$ mm)

(f) Legend: All plots

Figure 4.5: Comparative study - MCS vs Subset Simulation convergence. The x-axis for all plots shows labels for N = number of FEM simulations with scale $\frac{1}{\sqrt{N}}$. Shaded areas indicate 95% confidence interval regions for each sampling methodology (MCS, MH, CMH, Gibbs). N for Subset Simulation was calculated as a cumulative total for each $T_{i+1}|T_i$, including N for initial MCS step, T_1.

Figure 4.6 compares the convergence and simulation time for each sampling methodology. The relative efficiency of each sampling methodology is expressed in terms of the reduction in the width of the 95% confidence intervals versus simulation time. The average simulation time for each MCS analysis does not vary significantly as the sampler is likely to run many analyses in the more numerically stable (lower output displacement) regions of the sample space. More numerically stable samples require less computational resources for the solver to converge. By contrast, the MCMC samples for far from mean u_0 values sample more frequently from less numerically stable parts of the sample space and so each discrete FEM problem is likely to require more time to solve. In Figure 4.6, relative efficiency is calculated by taking half of the average 95% confidence interval width and multiplied by the normalised simulation time. Simulation times are normalised such that 1×10^5 MCS analyses have an average run time of 1 unit. Then, in Figure 4.6, higher efficiency is indicated by lower values on the y-axis as reducing confidence interval width and decreasing run time both lower the value on the y-axis.

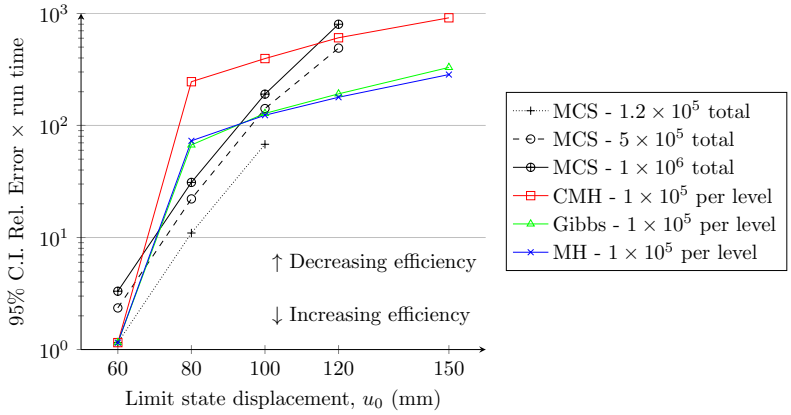


Figure 4.6: Comparative study - MCS and Subset Simulation reliability analysis comparison. Computational complexity is expressed as average confidence interval width times normalised average simulation time. Simulation times are normalised so that 1×10^5 MCS analyses have a time of 1 unit. Higher values on the y-axis indicate less efficient sampling methodologies.

4.3.4.3 Discussion

As in the verification study, the comparative study found that Subset Sampling was able to estimate far from mean responses more efficiently than MCS. However, MCS requires less computational effort to estimate $P(u > u_0)$ for

small u_0. Further, as in the verification study, it was found that Metropolis-Hastings sampling was the most efficient sampling methodology tested for Subset Simulation combined with nonlinear finite element analysis, followed by Gibbs and then Componentwise Metropolis-Hastings sampling.

With reference to Figures 4.4 and 4.5, all sampling methodologies converged to similar values for each u_0. However, the performance of MCS degraded rapidly and was not effective for estimating either $P(u > 120 \text{ mm})$ or $P(u > 150 \text{ mm})$. For Subset Simulation, MH and Gibbs sampling displayed very similar convergence rates. The slightly faster convergence of MH is likely because the Gibbs sampler may take larger jumps than the MH sampler and so would be more likely to jump to a sample that would be rejected for not having enough displacement to remain in the current minimum subset level. CMH exhibited slower convergence than the other methods. This was possibly because the CMH sampler was unable to mix through the state space as quickly as MH of CMH. The more oscillatory behaviour shown by CMH in Figure 4.5 suggests that a large amount of simulations were spent above the limit state followed by a large number of simulations below the limit state. Hence, although the CMH sampler did give similar mean estimates for $P(u > u_0)$ as the other samplers, the oscillations prevented convergence of the confidence intervals.

Figure 4.6 demonstrates that for $u_0 = 80$ mm and $u_0 = 100$ mm, MCS was more efficient than Subset Simulation. This is interesting because the probability $P(u > 100 \text{ mm})$ was found to be approximately 1×10^{-4} which is still quite small. This is approximately one order of magnitude greater than $\frac{1}{\sqrt{N}}$ for $N = 1 \times 10^5$ simulations, which was about the number of simulations used to find a relative accuracy of 1% for $P(u > 60 \text{ mm})$. For all $u_0 > 100$ mm, however, all Subset Simulation MCMC samplers were more efficient than direct MCS. Figure 4.6 additionally demonstrates that MH was found to be slightly more efficient than Gibbs sampling. The lower efficiency of CMH is likely because it accepts transitions more often than MH or Gibbs sampling and therefore spends a large number of FEM analyses computing far from mean (i.e. difficult to solve) FEM problems. Gibbs sampling is more likely to sample regions of the state space with low displacements than either MH or CMH. MH sampling, on the other hand, is more likely to spend a large number of cycles searching for a sample that will be accepted. In the context

of nonlinear FEM, each sample evaluation is very computationally expensive. It is therefore reasonable to expect that it would be more efficient to spend slightly more time searching for a suitable FEM sample before evaluating the sample, rather than accepting a very large number of samples (as in CMH). The discussion in §4.5 of [16] discusses how the transition acceptance probability, see equation (4.11), tends to zero as the dimension of the random samples vector goes to infinity.

The analyses in the comparative study suggest that as large a number as possible of probabilistic degrees of freedom in the random vector should be adjusted as a batch per iteration without altering so many degrees of freedom that the transition probability becomes vanishingly small.

4.4 Summary and conclusions

This Chapter discussed an algorithm, Subset Simulation, suitable for estimating very small probabilities of failure for probabilistic reliability problems with high dimensional, autocorrelated stochastic input spaces. Subset Simulation and Markov Chain Monte Carlo improves on direct Monte Carlo Simulation by reducing the number of discrete samples required to estimate the system reliability, without sacrificing the ability of direct MCS to find the critical failure mechanisms. Markov Chain Monte Carlo techniques suitable for use with Subset Simulation combined with nonlinear finite element analysis were presented. Further, a derivation of confidence intervals for the Subset Simulation mean estimate error probability distribution was presented.

The theoretical developments were tested numerically in Section 4.3. Both a linear and a nonlinear version of a probabilistic footing problem with spatially autocorrelated material property parameters were analysed by FEM. In the linear analysis, the performance of the Stochastic Finite Element Method was compared to direct Monte Carlo Simulation and Subset Simulation. While all three methods converged to similar values, Subset Simulation was found to be the most efficient technique for estimating far from mean responses. For the nonlinear analysis, an elastoplastic material model was introduced. Subset Simulation and Monte Carlo Simulation were compared and, again, Subset Simulation was found to be more efficient than Monte Carlo Simulation when estimating the far from mean structural response.

It is also worth noting that the efficiency assessment has been made by considering only a simple reliability quantity of interest (in particular, the threshold indicator function). If a more complicated reliability measure was used for each simulation, such as Factor of Safety, the performance of Subset Simulation relative to Monte Carlo for far from mean responses would be degraded. This is because the computational complexity of evaluating such a measure would increase as the degree of failure increased (for nonlinear models). The average complexity of evaluation of a Quantity of Interest will be fixed for a direct Monte Carlo simulation. As such, Subset Simulation may not be more efficient than direct Monte Carlo in some cases [193]. Investigation of these issues would be an interesting avenue for future work.

Three different Markov Chain Monte Carlo samplers were tested. Specifically, the performance of Metropolis-Hastings, Gibbs and Componentwise Metropolis-Hastings were compared on the numerical problems above. Metropolis-Hastings was found to be the most effective MCMC sampler used in all analyses, followed by Gibbs and then by Componentwise Metropolis-Hastings sampling.

Improved Markov Chain Monte Carlo techniques may also be interesting avenues for future research. As Metropolis-Hastings was found to be more efficient for nonlinear finite element problems than either of the componentwise samplers tested, the number of potential MCMC samplers available for Subset Simulation increases. In particular, Hamiltonian Monte Carlo techniques may possibly improve the rate of convergence for subset level simulations [55]. Methods to reduce the rejections during subset sampling, outlined in [290], would also potentially be useful.

A further interesting avenue for future work would be a more detailed consideration of the influence of spatially random fields on the positive-definiteness of elastic strain and plastic dissipation energies. The discretisation of spatially autocorrelated random fields on the level of finite element mesh integration points serves to effectively filter out high frequency field fluctuations. To address positive definiteness issues, the scale of fluctuations of random field models of material parameters would have to be considered. In particular, it would be interesting to analyse the impact of the choice of random field discretisation (basis functions and quadrature points) on pointwise energy constraints. This could potentially be assessed using random field threshold

crossing probabilities, as discussed in [5], as a means to explore the likelihood of material parameter fluctuations into values that would cause inadmissible energy states.

By combining Subset Simulation with the Random Finite Element Method, it is possible to compute small probabilities of failure for complex problems. The computational complexity limits imposed by the solution of the finite element problems are still, however, challenging. It will be necessary to apply advances in computational techniques and numerical methods to address stochastic problems of increasing complexity.

Chapter 4 Appendix: Proof of equation (4.19) - a k-fold product distribution density function equation

The proof of equation (4.19) proceeds by a simple induction argument. The probability of the product of two random variables, $Z = XY$ is given in equation (4.18) as, after reorganising:

$$f^Z(z) = \int_{-\infty}^{\infty} f^Y\left(\frac{z}{x}\right)\frac{1}{|x|}f^X(x)dx \tag{4.25}$$

where $f^X(x)$, $f^Y(y)$ and $f^Z(z)$ are probability density functions for X, Y, Z respectively.

Let Z_k be the product distribution formed from random variables X_i so that $Z_k = \prod_{i=1}^{k} X_i$. Then $Z_1 = X_1$ and $Z_i = X_i Z_{i-1}$ for $1 < i \leq k$. Let the density function for any Z_i be $f_i^Z(z_i)$ and the density function for any X_i be $f_i^X(x_i)$. Also, for $i = 1$, $f^{Z_1}(z_1) = f^{X_1}(x_1)$. It will be shown that the product probability density for k distributions can be written, for $k \geq 2$:

$$\begin{aligned} f_k^Z(z_k) = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f_k^X\left(\frac{z_k}{w_{k-1}}\right)\left[\prod_{i=2}^{k-1} f_i^X\left(\frac{w_i}{w_{i-1}}\right)\frac{1}{|w_{i-1}|}\right] \\ \times \frac{1}{|w_{k-1}|}f_1^X(w_1)dw_1\ldots dw_{k-1} \end{aligned} \tag{4.26}$$

Starting with the base case, for $k = 2$, equation (4.25) gives:

$$f_{2,\star}^Z(z_2) = \int_{-\infty}^{\infty} f_2^X\left(\frac{z_2}{w_1}\right)\frac{1}{|w_1|}f_1^X(w_1)dw_1 \tag{4.27}$$

where x_1 has been re-written as w_1. Expanding equation (4.26) for $k = 2$ gives:

$$f_2^Z(z_2) = \int_{-\infty}^{\infty} f_2^X\left(\frac{z_2}{w_1}\right)\frac{1}{|w_1|}f_1^X(w_1)dw_1 \tag{4.28}$$

Note that the terms in the product $\prod_{i=2}^{k-1}$ do not appear in equation (4.28) because $(k-1=2-1) < (i=2)$. Then, from equations (4.27) and (4.28), $f_{2,\star}^Z(z_2) = f_2^Z(z_2)$ so equation (4.26) is true for $k = 2$.

Next, assume equation (4.26) is true for $k = j$. Then, for $k = j+1$:

$$\begin{aligned} f_{j+1}^Z(z_{j+1}) = &\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f_{j+1}^X\left(\frac{z_{j+1}}{w_j}\right)\left[\prod_{i=2}^{j} f_i^X\left(\frac{w_i}{w_{i-1}}\right)\frac{1}{|w_{i-1}|}\right] \\ &\times \frac{1}{|w_j|}f_1^X(w_1)dw_1\ldots dw_j \end{aligned}$$

and, using equation (4.25):

$$f_{j+1,\star}^Z(z_{j+1}) = \int_{-\infty}^{\infty} f_{j+1}^X\left(\frac{z_{j+1}}{w_j}\right)\frac{1}{|w_j|}f_j^Z\left(w_j\right)dw_j \tag{4.29}$$

Using the induction hypothesis that equation (4.26) is true for $k = j$, equation (4.29) can be written:

$$\begin{aligned} f_{j+1,\star}^Z(z_{j+1}) = &\int_{-\infty}^{\infty} f_{j+1}^X\left(\frac{z_{j+1}}{w_j}\right)\frac{1}{|w_j|} \\ &\times \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f_j^X\left(\frac{w_j}{w_{j-1}}\right)\left[\prod_{i=2}^{j-1} f_i^X\left(\frac{w_i}{w_{i-1}}\right)\frac{1}{|w_{i-1}|}\right] \\ &\times \frac{1}{|w_{j-1}|}f_1^X(w_1)dw_1\ldots dw_{j-1}dw_j \end{aligned}$$

Reorganising:

$$\begin{aligned} f_{j+1,\star}^Z(z_{j+1}) = &\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f_{j+1}^X\left(\frac{z_{j+1}}{w_j}\right) \\ &\times\left[\prod_{i=2}^{j-1} f_i^X\left(\frac{w_i}{w_{i-1}}\right)\frac{1}{|w_{i-1}|}\right] f_j^X\left(\frac{w_j}{w_{j-1}}\right)\frac{1}{|w_{j-1}|}\frac{1}{|w_j|}f_1^X(w_1)dw_1\ldots dw_j \end{aligned}$$

Finally, collecting terms:

$$f^Z_{j+1,\star}(z_{j+1}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f^X_{j+1}\left(\frac{z_{j+1}}{w_j}\right) \left[\prod_{i=2}^{j} f^X_i\left(\frac{w_i}{w_{i-1}}\right) \frac{1}{|w_{i-1}|}\right] \times \frac{1}{|w_j|} f^X_1(w_1) dw_1 \ldots dw_j \tag{4.30}$$

From equations (4.29) and (4.30), $f^Z_{j+1,\star}(z_{j+1}) = f^Z_{j+1}(z_{j+1})$. Then, as equation (4.26) is true for the base case $k = 2$ and for the inductive case $k = j + 1$, equation (4.26) is true for all $k \in \mathbb{N}, k \geq 2$ by induction.

Chapter 5

Deep Artificial Neural Network Surrogate Models for Partial Differential Equation Uncertainty Quantification

Contents

Figures

Tables

Chapter 5 Overview

Key developments in Chapter 5 include:

- Section 5.2 provides background material on Deep Learning with Artificial Neural Networks (ANNs), including both feedforward and Recurrent Neural Networks (RNNs).
- Section 5.3 details an original contribution: a method for using ANNs as surrogate models for complicated numerical PDE solvers, for example Finite Element Methods, for both time dependent and independent problems. The proposed surrogate model method is used to accelerate the convergence of Monte Carlo based Uncertainty Quantification.
- Section 5.3.4 studies the theoretical computational complexity of the proposed Monte Carlo simulation acceleration method.
- Section 5.4 presents a number of case study analyses. A linear random boundary value problem and nonlinear initial value scalar PDE problem is analysed. Suitable training schemes and architectures for ANN PDE surrogates are demonstrated.

5.1 Introduction

This Chapter will demonstrate that modern Machine Learning techniques can be used to great effect for augmenting Uncertainty Quantification for physical systems modelled by Partial Differential Equations (PDEs). Uncertainty Quantification will refer to estimating probability densities and the expected value of Quantities of Interest over the space of outputs of some Stochastic PDE for a given distribution of inputs. Probabilistic problems of this sort require the solution of computationally demanding integrals. First, the high dimensionality of the discretisations used to approximate PDE problems presents challenges, necessitating the use of specialised integration methods. Second, any numerical integration method requires multiple evaluations of the function to be integrated. For probabilistic PDE problems, these function evaluations can be computationally demanding. To mitigate this, regression methods can and have been used as so-called surrogate models. Surrogate models attempt to infer unknown function values based on known values.

Given a surrogate model with low error, it is possible to perform fast high dimensional numerical integration for Uncertainty Quantification by repeatedly evaluating the surrogate, rather than the full PDE equations.

In this Chapter, it is shown that modern incarnations of Artificial Neural Networks (ANNs) can be used as highly effective surrogate models for probabilistic PDE problems. In particular, it is demonstrated that by the use of appropriate activation functions the training issues that have typically plagued earlier attempted applications of deep Multilayer Perceptron Models can be avoided. Feedforward Neural Networks can be used effectively to model boundary value PDE problems and fixed time outputs for initial value problems. Recurrent Neural Networks can be used for initial value problem PDE time sequence prediction. This is demonstrated by three numerical test case problems based on a finite element analysis of a linear Poisson problem and nonlinear, time dependent heat equation with uncertain inputs modelled by spatially autocorrelated random fields. The results indicate indicate that direct Monte Carlo Simulation based Uncertainty Quantification can be improved upon by introducing Neural Networks as surrogate models for the PDE equations. This Chapter also demonstrates effective techniques for designing and training these ANN surrogate models. Further, an overview meta-analysis of the time complexity of surrogate modelling in a supervised learning context is presented.

With reference to §18 of [323], learning can be thought of as a process which asks some system to generate a predictive model of its environment. This Chapter focuses on how techniques from *supervised learning* can be leveraged for PDE Uncertainty Quantification with specific reference to ANNs. From §18 of [323] and [122], in supervised learning an agent or algorithm attempts to infer a function from labelled data. The labelled data, or *training examples*, consist of pairs of inputs and the corresponding outputs to a function. The goal of the learning process is to use the training examples to infer the structure of the function that maps from inputs to outputs such that the error when generalising from the training examples to an unseen input is minimised. When the classification function to be learnt is real valued, the supervised learning task is referred to as regression. This Chapter demonstrates the use of ANNs as adaptive surrogate models for accelerating sampling based Uncertainty Quantification. The supervised learning approach to Uncertainty Quantification requires that a deterministic PDE solver is available for the computation in

order to collect training samples. The goal of the learner is to optimally generalise across sampled solution instances and learn a function that maps from PDE inputs to PDE outputs. This surrogate model can then be used to for Uncertainty Quantification for a given input distribution.

Polynomial Chaos Expansions (PCE) [351] and Support Vector Regression (SVR) [323, 81] are closely related forms of supervised learning. In particular, SVR with a polynomial kernel is very similar to PCE methods. Both of these techniques leverage projections onto high dimensional polynomials to represent nonlinear functions for regression over some space via Mercer's Theorem [356, 253, 210, 256, 254, 351]. Polynomial Chaos approaches can be seen as an alternative to explicit specification of a SVM polynomial kernel [328]. Instead, polynomial feature functions are constructed using orthogonal polynomial sequences such as Hermite polynomials [141]. For other polynomial series choices see [399, 356]. It is important to note that the optimisation objective is often calculated differently for Polynomial Chaos based methods when compared to what is used for Support Vector Regression [343]. Regression features are the type of functions used to represent the surrogate model and the interactions between these features. PDE Uncertainty Quantification by PCE and SVR are limited in part because of the need to define the regression features. Manual feature engineering becomes difficult for complicated high dimensional functions. By using ANNs, the feature functions do not have to be designed by hand. Rather, the challenge is to design the ANN architecture.

Recently, Artificial Neural Networks (ANNs) have seem dramatic performance gains and have become the dominant method for supervised learning. This is in part due to the success of deep learning methods which have resolved some of the issues with traditional Multilayer Perceptron networks. An extensive review of the recent history of ANNs is given in [327]. Following from these developments, this Chapter demonstrates that ANNs can be used for PDE Uncertainty Quantification. The use of ANNs for this purpose has long been known, for example see [312] and the discussion in §4.5.6 in [359]. The waning popularity of ANNs in by the late 1990's (see [232] and discussion below for more details) is mirrored in the relative lack of publications between that period and now. Techniques to mitigate the feature engineering problem for surrogate models based on adaptive response surfaces computation (see [42, 47, 48, 357]) became the dominant method. Recently, however, with the

resurgence of ANNs across many fields have led to new research into Uncertainty Quantification using ANNs [7, 211, 209].

ANNs are effective at learning adaptive basis function representations of functions [344]. The numerical experiments in this Chapter demonstrate that, by sensible architecture choices, ANNs can be used as accurate surrogate models. In particular, it is demonstrated that deep networks (both in space and time) are well suited to Uncertainty Quantification regression tasks. Rather than using the traditional sigmoid ANN activation functions as in Multilayer Perceptron networks [173], rectified units (such as ReLU and ELU functions [276, 77]) can be used to increase the efficacy of deep networks. Recurrent Neural Networks (in the context of this Chapter) model sequences of outputs for a given PDE input. Long Short-Term Memory cells [186] are shown to be a useful addition for PDE time series prediction using Recurrent Neural Networks. Unfortunately, the design of ANN architectures is still, itself, challenging and a current area of research [258, 388, 151]. Trial and error is one approach to architecture design and is the method adopted in this Chapter. By demonstrating that ANNs can be used as effective surrogate models, this Chapter suggests that the more complicated problem of automated architecture discovery would be a worthwhile avenue for future research.

Section 5.4 presents three numerical experiments to support the claims of this Chapter. The numerical experiments are based on finite element solutions of a scalar diffusion PDE. First, a probabilistic linear Poisson equation is analysed. The input diffusion coefficient and source fields are modelled by spatially autocorrelated random fields. The solution space probability distribution is estimated by a Kernel Density approximation. The performance of Monte Carlo Simulation and ANN based surrogate model augmentation techniques for estimating the solution space density are compared. Several different ANN architectures are analysed. It is shown that deep, rectified ANNs outperform direct Monte Carlo analysis for the experiments presented. Next, time dependency and nonlinearity are introduced by extending the first problem into a heat equation with a nonlinear material diffusion function. An additional random field representing the unknown initial condition is introduced. This experiment compares the performance of two types of ANN to Monte Carlo Simulation for estimating the output probability at a fixed time. It is again shown that the deep rectified ANN has the best performance. Suggestions

for an appropriate training schedule are also discussed. Finally, the third numerical experiment demonstrates that Long Short-Term Memory Recurrent Neural Networks can be used as surrogate models for predicting, given an input, the entire sequence of outputs of the nonlinear heat equation used for the second experiment. These numerical experiments demonstrate and discuss effective ANN and RNN design for PDE Uncertainty Quantification and demonstrates that these networks can be used as accuracy surrogate models for both boundary and initial value problems. Although more complicated analyses, such as time-dependent random forcing, were not considered the proof-of-concept analysis presented in this Chapter are promising. The extension of the proposed method to more complex and realistic problems would be a useful area of future research.

5.2 Supervised learning

Supervised learning can be subdivided into *classification* and *regression* [323]. Classification refers to learning problems where the space of outputs from the function of interest is a finite set and the output for any given input is one of these values. A critical relevant example for Uncertainty Quantification is the reliability analysis problem [30, 157]. The unacceptable performance indicator function applied to the output space of a Stochastic PDE reduces each output from the PDE to one of two classes, 0 or 1 (i.e. 'no failure' or 'failure' respectively). A "good" supervised learning classifier for the reliability analysis problem would learn to correctly assign the label 0 or 1 to each possible input. By contrast, regression refers to learning problems where the proposed output value is some number or vector. As such, the principal difference between regression (in the real number case) and classification is that the probability to find the exact right value in a regression problem is zero because the probability of selecting a precise value from a continuous random variable is zero (see §1.6 of [61]). An Uncertainty Quantification based example for regression would be to infer the correct solution of a PDE given a particular input.

This Section gives formal definitions for supervised learning problems. Following this, the Artificial Neural Networks approach to supervised learning is detailed. By contrast, the popular Polynomial Chaos based approaches to

PDE Uncertainty Quantification [399, 357, 356, 141, 142] can be viewed as a form of Support Vector Regression using particular types of polynomial kernel [188]. Rather than explicitly define a polynomial kernel as in Support Vectors methods [67], Polynomial Chaos methods build the polynomials using predefined series expansion polynomial sequences. The end effect, regression in a space with higher dimensionality than the input space, is the same. The difference is in the method used to build the polynomial representations. ANNs have two major advantages over SVMs. First, there is no need to explicitly define feature kernels. Second, deep ANNs can be used to automatically generate highly nonlinear representations of the function to be learnt without relying on human designers hand crafting complicated representations. This Section details the general aspects of ANN models that are useful for the Uncertainty Quantification specific developments in Section 5.3 and the numerical experiments in Section 5.4. In particular, deep feedforward and recurrent networks, as well as training algorithms suitable for deep networks, are described.

5.2.1 Supervised learning definitions

The goal of the supervised learning task is (§18.2 [323], §2 [79] and [97]) to infer a function, h, from the function space $\mathcal{H}$ (the *hypothesis space*) that is the best approximation to the unknown function $f : \mathbf{X} \to \mathbf{Y}$. f is a map from the *input space*, $\mathbf{X}$ to the *output space*, $\mathbf{Y}$. The approximation of f using h is made given a *training set*, $\mathcal{T}_N(\tau)$ for $\tau = (x, y)$, of N input-output pairs:

$$\mathcal{T}_N = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \in (\mathbf{X} \times \mathbf{Y})^N$$

where each $x_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}$ forms a pair $\tau_i = (x_i, y_i)$. Additionally, each y_i is generated by the unknown function $f(x_i)$. Note that the pairs in the training set are ordered pairs, that is, $(a, b) \neq (b, a)$.

A *test set* can be withheld from the training set and used to evaluate the performance of h in approximating f. The generalisation error on novel examples is quantified using a functional $L : \mathbf{Y} \times \mathbf{Y} \to \mathbb{R}^+$. The function $L(h(x), y)$ is known by several names including the *loss functional* (§10 [22]), *loss function* or the *cost function* (§2 [79]). For a general regression problem, the best hy-

pothesis is selected such that the expected value of the loss is minimised:

$$h^*(x) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbb{E}\left[L[h(x), y]\right] \tag{5.1}$$

A typical loss functional for regression is some form of mean squared error of $h(x)$, for example:

$$L[h(x), y] = \frac{1}{2}\|h(x) - y\|^2 \tag{5.2}$$

By contrast, the function to be learnt may be a probability distribution. In this case, the training examples are assumed to have been sampled according to $f(x, y) = P(x, y)$. Learning, in this case, can be categorised as either *discriminative* or *generative* learning [68]. In discriminative modelling, the goal is to estimate the conditional probability $h(x) = P(y|x)$. In generative modelling, the goal is to estimate the probability $h(x, y) = P(x, y)$.

For a given loss functional and h, the *risk* $R[h]$ is the expected loss of h over the training set:

$$R[h] = \mathbb{E}\left[L(h(x), y)\right] = \int L[h(x), y] dP_{\mathcal{T}}(\tau = (x, y)) \tag{5.3}$$

The risk here is distinct from the risk associated with other usages common in Uncertainty Quantification (for example, the probability of and consequences due to a building collapsing). The intended usage of the term risk should be clear from the context. The goal of learning is to find $h^* \in \mathcal{H}$ such that the risk is minimised. As the generating distribution $P(x, y)$ (in other words, the function f) is unknown, the risk must be estimated by the empirical risk. The empirical risk estimates the true $R[h]$ using the training data $\mathcal{T}$:

$$R[h] \approx \hat{R}[h] = \frac{1}{N} \sum_i L(h(x_i), y_i) \tag{5.4}$$

Supervised learning attempts to approximately learn the unknown function f by minimising the empirical risk:

$$\hat{h}^*(x) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \hat{R}[h] \tag{5.5}$$

The foundational assumption of supervised learning is that it is possible to infer information about the unknown function f using only partial information

from the training samples. Quoting §2.2.2 of [269]:

> **The inductive learning hypothesis.** Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

However, it is important to note that there is a trade-off between the expressiveness of a hypothesis space (range of functions within the space) and the complexity of finding a good hypothesis within that space (§18.2 [323]). Using too large a hypothesis renders search within the space difficult. Different supervised learning techniques, such as ANNs and kernel methods, use different restrictions on the possible hypothesis spaces and different search techniques within these spaces.

The choices made when setting up a supervised learning task, such as the choice of a hypothesis space and the choice of a loss functional, effectively encode prior beliefs of the human designers into the task. These priors may help an algorithm perform well on a machine learning task by, for example, restricting the hypothesis search space to a smaller set. Conversely, a poor encoding of prior beliefs may hurt the performance of a machine learning algorithm by oversimplifying a problem or being unable to effectively capture the true relationships within the data. These issues are discussed in detail in the review publication [37]. The central prior belief that is encoded in the majority of supervised learning methods is that of *smoothness*. That is, for a function f to be learnt, if $x \approx y$ then $f(x) \approx f(y)$. Techniques that require smoothness to work, such as linear regression and kernel (or basis) learning are not effective when this assumption is not true.

To improve learning performance a *regularisation* term, $C[h]$, is often applied to the empirical risk such that the learning problem is to find:

$$\hat{h}^*(x) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \hat{R}[h] + \lambda C[h] \tag{5.6}$$

where λ is a constant controlling the strength of the regularisation. In this case, empirical risk minimisation may be referred to as regularised empirical risk minimisation or as *structural risk minimisation* [383, 384]. One of the goals of regularisation is to improve the generalisation performance of fitted models by favouring certain functions more than others. There are a num-

ber of interpretations of regularisation terms that appear in the literature. Intuitively, regularisation terms are often applied to favour simpler functions over more complex ones. This can greatly improve the performance on fitting data with many irrelevant features [281, 282]. Regularisation can be understood, directly in a Bayesian sense as a selection of a particular prior over the hypothesis space, $P(h)$, where $P(h|data) \propto P(data|h)P(h)$ [121, 206].

Commonly, Tikhonov regularisation [372, 373] is used for this purpose. Tikhonov regularisation can be applied to ill-posed problems (see [164]) which frequently arise in the solution of inverse problems, particularly when finite computational precision is used. For example, even if there is a unique solution to $Ap = q$ for some operator A and unknown function q with functions $p, q \in Q$, finding p will be ill-posed if there are numerical stability problems present when calculating a solution. Specifically, for an unstable problem $\|q - q_\delta\| < \epsilon$ for small ϵ then $\|p - p_\delta\|$ will be large. In this case, risk minimisation on p_δ will fail to find a hypothesis close to f even if ϵ approaches zero. With the addition of a regularisation term, the minimisation of the regularised risk will be able to converge to the correct solution. Detailed discussion is presented in [383]. An alternative method for regularisation called *dropout*, which averages the results of several approximate models by disabling links within a connectionist model, has shown to work well and is discussed in [350].

5.2.2 Artificial Neural Networks

An Artificial Neural Network (ANN), in very general terms, computes some function by representing a computation as a graph. A history of the early development of Artificial Neural Networks is given in §18.7 of [323]. Later developments are described in [151]. The description here follows from [151] and §18 in [323]. The state of the nodes of the graph is dependent on the weighted sum of values stored connected graph vertices. Each vertex of the graph is called a *node* or a *unit*. The links between units i and j are assigned a numeric weight, w_{ij}. Units output *activation* values, denoted a_i for unit i.

The input to a node with n incoming links is calculated as a weighted sum of the incoming activations:

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i \tag{5.7}$$

The activation with index 0 is called the *bias*. Bias inputs to a unit are typically always activated and help units to compute translations of the input [173].

Given the input to a unit, in_j, the unit applies an *activation function*, $\sigma(in)$:

$$a_j = \sigma(in_j) = \sigma\left(\sum_{i=0}^{n} w_{i,j} a_i\right) \tag{5.8}$$

Activation functions are defined in more detail in Section 5.2.2.2.

There are a number of ANN architectures that specify different connection graphs. Two architectures, feedforward and Recurrent Neural Networks are discussed in this Chapter in Sections 5.2.2.1 and 5.2.2.3 respectively. Other types of architectures include Restricted Boltzmann Machines [180, 184], Helmholtz machines [182], and Echo-State Networks [199]. The choice of graph architecture influences the training algorithm. Training algorithms for Neural Networks (typically) find the values of the weights required to ensure the calculation performed by the Neural Network is suitable. In a supervised learning regression setting, the training objective is to minimise the error when estimating the value of a function for a given input by adjusting the connection weights. The backpropagation training algorithm is suitable for directed acyclic graph networks. The feedforward and recurrent architectures are appropriate for regression. In particular, it can be shown that these network architectures are capable of universal function approximation [85, 192] and have the theoretical capacity to represent essentially any relevant function.

Actually achieving accurate representations in practice is, however, challenging. Deep ANN architectures are particularly interesting because they learn hierarchical function approximations, with more abstract features learnt in deeper layers [223, 400]. By structuring function approximations in this way, information can be encoded very efficiently. Following from the discussion and review in §6.4 [151], although these advantages of deep networks have been anticipated for several decades [44], it has only recently become possible to train such networks effectively. Wide, shallow networks on the other hand were historically favoured as they could be trained and satisfied the universal approximation theorems [85, 192]. However, the width of the single layer network required increases exponentially with the size of the problem. Deep networks can avoid this problem. This Chapter demonstrates that deep networks can

be used effectively as function approximation surrogates for PDE Uncertainty Quantification problems. The necessary ANN definitions and discussion are presented in this Section.

5.2.2.1 Feedforward networks

The feedforward architecture organises ANN connections in a directed acyclic graph, that is, the connections are only in one direction. An input is fed into one end of the network and and an output is calculated at the other end. There is no internal state dependency on previous inputs or outputs as the flow of information in the network is unidirectional. Only the weights represent the current network state. It will be shown in Sections 5.3.3 and 5.4.2 that the feedforward architecture is well suited as a surrogate model for boundary value Partial Differential Equation problems. Further, it will also be shown that the feedforward model can be used for initial value problem prediction by, for example, prediction of the solutions of the initial value problem at some fixed time.

Feedforward networks are typically organised into several layers. Commonly, these layers consist of an *input layer*, *output layer* and a number of intermediate *hidden layers*. The advantage of a layerwise architecture is that the graph problem is vectorised. The incoming activations to all the units in a layer, as in equation (5.7), can be calculated by matrix multiplication. The outgoing activations from each unit in a layer, as in equation (5.8), are calculated for the result vector from the equation (5.7) matrix multiplication. The layerwise architecture is also useful for implementing the backpropagation training algorithm, described in Section 5.2.2.5.

Although there is no formal definition of a deep network, a deep feedforward network may be taken to be 3 or more layers. The number of layers for large neural networks, such as those for image classification, can be very high [223]. Networks with over 1000 layers have successfully been applied to supervised learning benchmarks [174]. Figure 5.1 demonstrates the feedforward architecture for three *densely connected* layers. Dense connections between layers connect each unit in the input layer to each unit in the output layer. By contrast, sparse connections set a number of the dense connections to zero [170]. Only densely connected layers are considered in this Chapter.

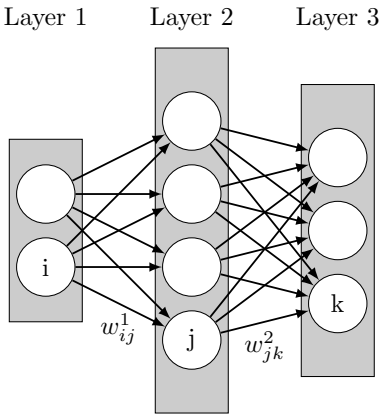


Figure 5.1: Three layer Feedforward Artificial Neural Network architecture showing units (circles), weighted connections (arrows). In the diagram, unit i in Layer 1 feeds forward to unit j in Layer 2 by weight w_{ij}^1. Similarly, j feeds forward to k in layer three by weight w_{jk}^2. Weight superscripts refer to the owning layer. Layers are shown as grey boxes surrounding the contained units. Densely connected weights are shown, that is, each unit the layer feeds forward all units in the next Layer. For the network shown, Layer 1 is an input layer, Layer 2 is a hidden layer and Layer 3 is an output layer.

5.2.2.2 Activation functions

ANNs are able to represent nonlinear functions by using nonlinear activation functions. It was discovered that part of the difficulty in training deep networks generally was related to the choice of activation function and the associated gradients of these functions (see Section 5.2.2.5). In this Chapter, it is shown that using *rectifier* units improves ANN training. Five types of activation functions are considered within:

$$\text{Linear: } \sigma(x) = x \tag{5.9}$$

$$\text{Sigmoid: } \sigma(x) = \frac{1}{1 + \exp(-x)} \tag{5.10}$$

$$\text{Tanh: } \sigma(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{5.11}$$

$$\text{ReLU: } \sigma(x) = \max(0, x) \tag{5.12}$$

$$\text{ELU: } \sigma(x) = \begin{cases} x & \text{if } x \geq 0 \\ a\left(\exp(x) - 1\right) & \text{otherwise} \end{cases} \tag{5.13}$$

The sigmoid, tanh, Rectified Linear Unit (ReLU) and Exponential Linear Unit (ELU) functions are shown in Figure 5.2. Linear units are used in this Chapter to allow for regression ANNs to output convenient real values with a suitable function range. Sigmoid and tanh units have a similar shape, but tanh units are known to have some desirable numerical properties relating to the scale of the function gradients when compared to sigmoid units [234]. ReLU and ELU units are examples of rectifier units. Note that the parameter a in equation (5.13) must be set. A constant value of $a = 1.0$ is used for all analyses in this Chapter. The main feature of rectifier units is the shape of the first derivative which is (roughly) zero over part of the domain and constant or linear (approximately) for the rest of the function domain [77].

The traditional ANN architecture is the *Multilayer Perceptron network* [173], a feedforward ANN with sigmoid units. These networks were found to to be hard to train when using a large number of layers [147, 327] due to the vanishing/exploding gradient problem. This is discussed in detail in Section 5.2.2.5. Further, Section 5.2.2.5 discusses how rectifier units (such as the ReLU and ELU) functions were found to be useful for mitigating the vanishing/exploding gradient problem [276]. In particular the piecewise linearity of rectified units helps to prevent repeatedly multiplied gradients from becoming too large or small [77]. ReLU and ELU units were found to be more effective than sigmoid units for the numerical experiments presented in Section 5.4.

5.2.2.3 Recurrent Neural Networks

By contrast with feedforward ANNs, Recurrent Neural Networks (RNNs) have loops in their graph structure. RNNs are dynamical systems and the response of the network to a given input depends on the previous inputs (§18.7, [323]). As such, RNNs are able to use memory of previously seen inputs to make predictions in the present. A simple RNN architecture is shown in Figure 5.3. Further, RNNs can be used to generate sequences [364, 153]. This can be achieved, for example, by taking the RNN to represent a time dependent function.

Following [154], the equations for a standard RNN are as follows. Let $i = (i_1, \cdots i_T)$ be termed the input sequence. An RNN computes the hidden sequence $h = (h_1, c \ldots, h_T)$ and the output vector sequence $v = (v_1, \cdots, v_T)$ by

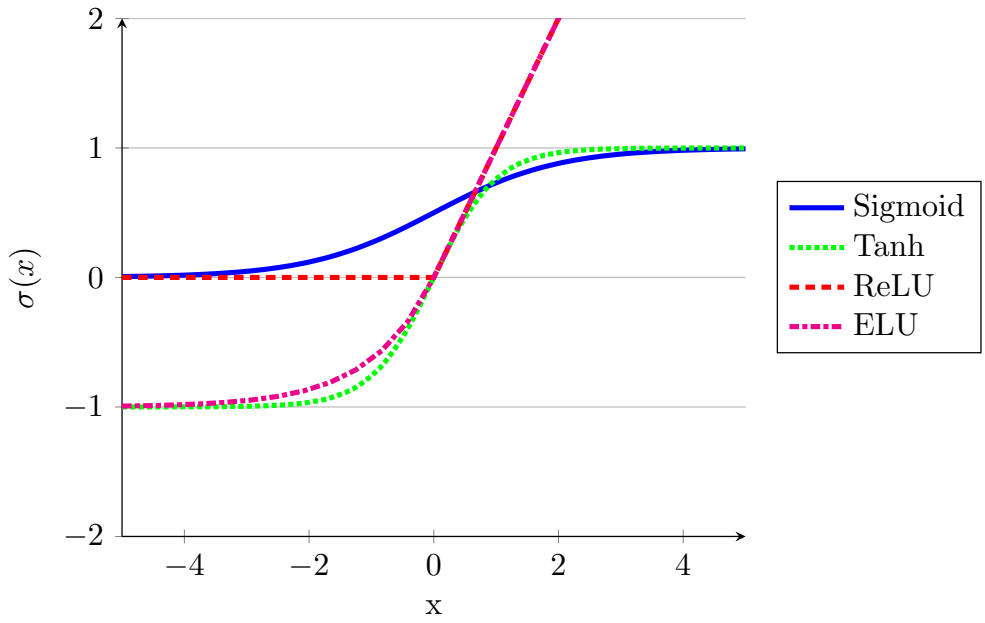


Figure 5.2: Neural Network activation functions used in this Chapter. See equations (5.10), (5.11), (5.12) and (5.13) for function definitions. For equation (5.13), $a = 1$.

iterating (over $t = 1$ to T) the following equations:

$$h_t = H(w_{vh} i_t + w_{hh} h_{t-1})$$
$$y_t = w_{hy} h_t$$

where w_{ij} terms denote weight matrices between units i and j and H is the hidden layer function. Note that bias terms are included in the weight matrices as index zero.

For example, consider an RNN that has been trained to predict the next value in an ordered sequence of T vectors, v_t for $t = 0$ to $t = T$, so that the network outputs the next value in the sequence given an input vector. That is, the network will attempt to predict v_{t+1} given v_t. The current value of the hidden unit can be sent forward through time to itself and a new output predicted at the subsequent time step. This process can be repeated until a sequence of the desired length has been output by the RNN. This sequence prediction architecture is depicted in Figure 5.4. Note that Figure 5.4 presents a *one-to-many* RNN architecture. The Seq2Seq algorithm, for example, uses a many-to-many architecture [365]. A one-to-many architecture is used for the numerical experiment in Section 5.4.4.

Although the function H can simply be one of the activation functions discussed in Section 5.2.2.2, this has been shown to be problematic when training networks to learn long time dependencies [39, 364, 291]. This is discussed in more detail in Section 5.2.2.5. Briefly, error gradients are unable to be computed in a numerically stable fashion for deep-in-time RNNs using standard activation functions and as such training these networks was difficult. Long Short-Term Memory (LSTM) units have been shown to help overcome some of these difficulties. LSTM units use so-called memory cells to control the flow of information through the network. Following from [154], an LSTM RNN computes the following update functions:

$$p_t = \sigma(w_{hp} + w_{hp}h_{t-1} + w_{ci}c_{t-1}) \tag{5.14}$$

$$f_t = \sigma(w_{if} + w_{hf}h_{t-1} + w_{cf}c_{t-1}) \tag{5.15}$$

$$c_t = f_t c_{t-1} + p_t \tanh\left(w_{ic}i_t + w_{hc}h_{t-1}\right) \tag{5.16}$$

$$v_t = \sigma\left(w_{iv}i_t + w_{hv}h_{t-1} + w_{cv}c_t\right) \tag{5.17}$$

$$h_t = v_t \tanh(c_t) \tag{5.18}$$

where $\sigma(\cdot)$ is the sigmoid activation, p, f and v are the input, forget and output gates vectors and c is the cell activation vector. Each of these vectors has the same dimension as h. Further, each of the weight matrices, w_{ij}, are diagonal. See [186] for a more detailed description. The essence of the LSTM is that the input, output and forget gates are able to control the flow of gradients by storing temporal correlations over long time periods. The various gates allow temporal dependencies to be dropped or amplified in a differentiable manner such that training can be carried using backpropagation. Further, the LSTM function has a gradient that helps to prevent vanishing gradients (discussed further in Section 5.2.2.5) and as such LSTMs are useful for deep-in-time networks. LSTM units are used for this purpose in the numerical experiments in Section 5.4.4.

5.2.2.4 Training directed acyclic ANNs by Stochastic Gradient Descent

Consider the supervised learning problem defined in Section 5.2.1. The feedforward and recurrent architectures are suitable for regression. Feedforward networks use a directed acyclic graph as a connection pattern. Recurrent

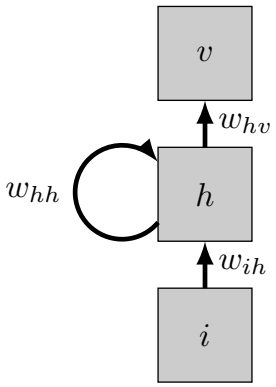


Figure 5.3: Schematic of simple Recurrent Neural Network architecture. Each of the layers i, h and v are connected by dense sets of weights, w_{ih} and w_{hv}. In addition to this feedforward architecture, layer h is connected to itself by a set of weights w_{hh}.

Neural Networks can be unfolded into a directed acyclic graph for training. Minimisation of the empirical risk (or expected loss across the training samples) using a given loss functional for discriminative learning is an optimisation problem. The most common approach to loss minimisation for ANNs is *Stochastic Gradient Descent* or SGD. Combined with the backpropagation algorithm, ANNs can be trained effectively.

Stochastic Gradient Descent (SGD) is an optimisation technique for finding the optimal function, $h(x)$, from the hypothesis space. SGD iteratively adjusts the current fitted hypothesis so as to minimise the error of $h(x)$ over a number of training examples. Let the hypothesis space $\mathcal{H}$ be parameterised by some vector θ. In the case of ANNs, the values of θ are the trainable weights in the network. With reference to equation (5.3), denote the ANN risk by $J(\theta) = R[h]$ so that θ represents the parameterisation of functions $h(x;\theta) \in \mathcal{H}$. Then for a given loss functional, $L[h(x;\theta), y]$, and training examples, y, denote the risk of some hypothesis function, $R[h(\theta)]$, by

$$J(\theta) = R[h(\theta)] = \int L[h(x;\theta), y] dP_{\mathcal{T}}(\tau) = \mathbb{E}\left[L[h(x;\theta), y]\right] \tag{5.19}$$

Stochastic Gradient Descent attempts to minimise the value of the loss functional by altering the parameters, θ, in a direction that will minimise the expected value of the loss given the training data. Denote the gradient of $J(\theta)$

with respect to θ by:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}\left[L[h(x;\theta), y]\right] \tag{5.20}$$

SGD iteratively updates the values θ in the direction that will decrease the empirical risk over *minibatches* (defined below):

$$\theta_{i+1} = \theta_i - \eta \nabla_{\theta_i} J(\theta_i) \tag{5.21}$$

where the parameter η is termed the *learning rate*. The learning rate controls the size of the jump θ takes in the direction of decreasing $J(\theta)$. Too low a learning rate and convergence to the optimal θ will be slow. Too high a learning rate and the training algorithm will oscillate, jumping over more optimal θ's or even causing the weights to diverge [181].

The gradient of the expected value of the loss is approximated by SGD using the loss gradients over subsets of the full training data set called minibatches (see §5 in [151]). Minibatch training can improve the efficiency of gradient descent algorithms [51, 242]. The approximate loss gradient over a minibatch of size n is:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \frac{1}{n} \sum_{i=1}^{n} L[h(x_i;\theta), y_i] \tag{5.22}$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L[h(x_i;\theta), y_i] \tag{5.23}$$

The SGD approach of estimating the loss gradient over minibatches is computationally much more efficient than the (non-stochastic) Gradient Descent method of using the full training set for every weight update [234]. Practically, SGD training is broken into a number of *epochs*. A single epoch updates θ by calculating the gradient $\nabla_\theta J(\theta)$ over each minibatch in the training set. This process is repeated for some number of epochs to progressively find the optimal $J(\theta)$.

As SGD attempts to greedily update the weights by always moving in the direction that will decrease the loss, it is a local optimisation algorithm [345]. This means that the initialisation values for the weights will impact on the final trained network as the starting position in weight space will influence the local minimum found by the SGD algorithm. Historically, attempts at

training deep networks were unsuccessful until the introduction of greedy layerwise pre-training [183, 38]. Further research suggested that the success of the layerwise method was due to the fact that it found useful starting locations in weight space [110]. More importantly, it was also found that greedy layerwise pre-training acted as an effective regulariser (in the sense discussed in Section 5.2.1) [36]. When training ANNs, using appropriate units (such as rectified activation functions) a similar regularisation effect can be achieved [350]. The deep ANNs in Section 5.4 of this Chapter were trained without using layerwise pre-training. Layerwise pre-training and rectified units also help to deal with the *vanishing gradient* problem discussed in Section 5.2.2.5.

Adaptive learning rate control has proven to be a significant factor in improved training for deep ANNs [36]. An important aspect of adaptive learning rate control algorithms is that they help to mitigate SGD local optimisation issues by avoiding convergence to poor local optima. There are a large number of competing SGD variants used for adaptive learning rate control including adaGrad [101], RMSProp and Adam [212]. RMSProp adapts the learning rate for each of the trainable parameters in the model, roughly, by dividing the learning rate for a particular weight by the mean gradient of recent weight updates. The Adam (Adaptive Moment estimation) algorithm extends RMSProp by including weight update factors based on the variance of recent weight updates. The Adam algorithm is used for the numerical analyses in Section 5.4 of this Chapter.

5.2.2.5 Backpropagation gradients for directed acyclic ANNs

The final component of ANN training is the calculation of the loss gradient with respect to θ. Backward propagation of errors or *backpropagation* is the primary technique used for calculating the required gradients for ANN Stochastic Gradient Descent when training networks with a directed acyclic graph structure. This algorithm evaluates the loss functional value at the output of the network for a given input and then passes these error values backwards through the network, accumulating the error at each unit. The gradient of the weights with respect to these errors can then be calculated. Backpropagation is simply the chain rule applied to the network function composition structure, see [45], §18 of [323] and §6 of [151]. A particularly useful derivation of the backpropagation equations is given in [44]. To compute the

layerwise weight gradients, using the error gradient, the first step is to compute the layerwise errors.

For a network that computes $h(x;\theta)$, let $J(\theta) = E(a;\theta)$ be the error at the network output given activations. Let the input to the network be written $a^1 = x$. The activations on layer l in the network can be written in terms of the layerwise activations, a^l, and activation functions $\sigma_l(\cdot)$:

$$a^l = \sigma_{l-1}(a^{l-1}) = \sigma_{l-1}\left(\sum_{i=0}^{n} w_{i,j}a_i\right) \tag{5.24}$$

The network output is the activation values at the final layer $(M-1)$:

$$h(x;\theta) = a^{M-1} \tag{5.25}$$

The error is computed in terms of the loss function (for example the sum of squared errors):

$$\nabla_\theta J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L[h(x_i;\theta), y_i]$$

where the loss at the final layer with respect to the final layer activations is given by:

$$a^M = L(a^{M-1};\theta)$$

For each layer, l, at each node i, the error derivative δ_i^l is calculated by passing layerwise errors backwards (from the output to the input) through the network. These errors can be used to compute the derivative of the loss with respect to the layer weights, θ^l.

$$\delta_i^l = \frac{\partial L}{\partial a_i^l} \tag{5.26}$$

$$\delta_i^l = \sum_{j=0}^{m} \frac{\partial L}{\partial a_j^{l+1}} \frac{\partial a_j^{l+1}}{\partial a_i^l} \tag{5.27}$$

$$\delta_i^l = \sum_{j=0}^{m} \delta_j^{l+1} \frac{\partial a_j^{l+1}}{\partial a_i^l} \tag{5.28}$$

where, at the final layer, $\delta^M = 1$.

Using the backward errors, δ, the gradient of the loss with respect to the layer weights, θ^l can be calculated by considering that the loss is a function of the output of layer l, a_j^{l+1}, which is itself a function of the layer weights:

$$a_j^{l+1} = \sigma_l \left(\sum_{i=0} \theta_{i,j}^l a_i^l \right) \tag{5.29}$$

so that at layer l the gradient of the weights, θ^l, with respect to the loss is given by:

$$\nabla_{\theta^l} L = \frac{\partial L}{\partial \theta^l} [a^{l+1}(\theta^l)] \tag{5.30}$$

$$\nabla_{\theta^l} L = \sum_{j=0}^{m} \frac{\partial L}{\partial a_j^{l+1}} \frac{\partial a_j^{l+1}}{\partial \theta^l} \tag{5.31}$$

$$\nabla_{\theta^l} L = \sum_{j=0}^{m} \delta_j^{l+1} \frac{\partial a_j^{l+1}}{\partial \theta^l} \tag{5.32}$$

These gradients are used to compute the gradient descent updates as in equation (5.21).

As the gradients of every function in the network must be computed for backpropagation, all functions within the network must be differentiable. This is particularly relevant when choosing activation functions so that the derivative $\partial a_j^{l+1} / \partial a_i^l$ can be computed. Other ANN training algorithms, such as the REINFORCE method often used for policy gradient algorithms [395, 366], do not necessarily require full network differentiability.

It has long been known that the so-called *vanishing gradient problem* was responsible for difficulties faced when training deep ANNs [146, 327]. When passing error gradients backwards through an ANN for backpropagation, the error gradients at each layer are multiplied. For certain activation functions, for example sigmoid units, the gradients are bounded. When the error is large, the gradients will be small. When the error gradients are small, it will take SGD a very large number of iterations to reach better weights. As the error gradients are multiplied together through each layer, the error gradients can vanish with depth through the network. Similarly, the *exploding gradient problem* occurs for units with unbounded gradients. The impact of the exploding gradient problem is essentially the same on the ANN as vanishing gradients in that the network will not converge in either case. By using rectified units (see

Section 5.2.2.2) the vanishing/exploding gradient problem can be alleviated in deep networks [276, 147]. Rectifier units achieve this, in part, by ensuring that the backpropagation gradients are of a reasonable (often linear) form.

These difficulties are magnified in RNNs when long time dependencies are modelled [39, 364, 291]. Although backpropagation can be applied directly to feedforward networks, Recurrent Neural Networks must be *unfolded in time*. This unfolding allows for the function represented by an RNN to be computed using a directed acyclic graph computation structure and is shown in Figure 5.4. This then allows for the appropriate gradients to be passed backwards through the network along the directed network links. This process is referred to as *Backpropagation Through Time* [392]. For RNNs, LSTM units (see Section 5.2.2.3), help to alleviate the vanishing gradient problem [186]. The LSTM effective gradient is either 0 or 1 meaning that gradients simply pass through (or are blocked) at the LSTM cells. This does not totally solve the vanishing gradient problem for LSTM networks, but is effective in practice [153].

Modern backpropagation implementations are typically based on Automatic Differentiation [307, 279], a technique for calculating function gradients in computer programs by considering the gradients of all functions evaluated when a program executes. Popular implementations used for ANN programming include TensorFlow [1] and Theano [371]. The use of Automatic Differentiation based backpropagation decreases development time as only simply function differentials need to be found a single time for inclusion in a program. Further, human error is reduced as difficult differentials of program routines do not need to be found manually.

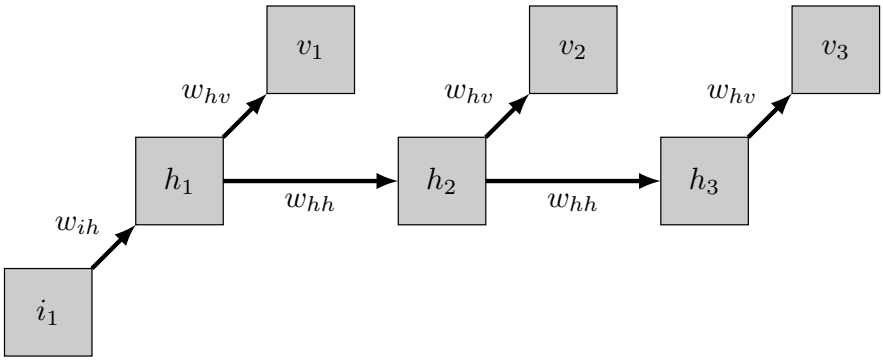


Figure 5.4: One-to-Many Recurrent Neural Network architecture after unfolding three steps deep in time. The input to this network is the vector (i_1) and the output is the sequence of vectors (v_1, v_2, v_3). Sequences for this network are generated by starting with an initial state, i_1 and then feeding forward through the network. Arrows indicate the flow of information through the network unit sets (depicted as boxes). w_{ab} labels refer to dense weight sets between unit sets a and b.

5.3 Supervised learning for Uncertainty Quantification

Artificial Neural Networks can be used as a nonlinear regression technique to estimate the output of deterministic PDE solvers. By explicitly considering how to optimally generalise from known data about the input and output spaces, it is possible to make estimates of the probabilities of output quantities of interest that are as accurate as possible in the minimum amount of time. ANNs are useful for this task as they can be shown to be universal function approximation tools [85, 192]. In particular, recent developments in deep learning have opened up the possibility for improved feature representation by ANNs [232]. In the language of Uncertainty Quantification, supervised learning can be used to build surrogate models to estimate model outputs for a given input. In this Section, a particular ANN approach to Uncertainty Quantification is described.

5.3.1 Uncertainty Quantification

Before continuing, it is useful to give a clear definition of Uncertainty Quantification [29, 318, 356] for PDE problems and the relationship to what is termed supervised learning in the Machine Learning literature. There are two main goals of Uncertainty Quantification. The first goal is to infer the probability measure induced on one space given a probability measure on another space and a measurable map between the two spaces. The second problem is to estimate the expected value of some function defined on the second space given the induced probability measure.

More formally, let $x \in X$ be the *input space* with points x and $y \in Y$ be the *output space* with points y. Let $y = f(x)$ be a measurable map (as per the definition in §18 in [168]). Let $P_X(x)$ be the *input space distribution*. Both $f(x)$ and $P(X)$ are given as a part of the problem specification. The goal is to infer $P_Y(y)$, the *output space distribution* induced on Y by the map $f(x)$ and the pushforward measure (see §3 and §9 of [50], Proposition 3.2.1 from [219] and [244]):

$$P_Y(y) = P_X(f^{-1}(y)) \tag{5.33}$$

The second problem of Uncertainty Quantification is to estimate *Quantities of Interest* (QoI). These are defined in this Chapter as the expected value of some function, $g(y)$, of the output space, given the output space distribution:

$$\mathbb{E}\left[g(y)\right] = \int_Y g(y) P_Y(y) dy \tag{5.34}$$

When considering Uncertainty Quantification for Partial Differential Equations, $f(x)$ represents the map from inputs to solutions defined by the PDE. $P_X(X)$ typically represents some distribution over the possible inputs to the PDE solver. It is typically not possible to exactly solve and invert $f(x)$ and therefore numerical approximations must be used. There are several forms that these approximations may take, however, a typical example of an approximation process is the Finite Element Method and the numerical variational problems solved by this technique. In this Chapter, the Finite Element Method will be used for the numerical analyses in Section 5.4. For these probabilistic PDE numerical analyses, the focus is on the first Uncertainty Quantification problem, estimating the output space probability density.

Integration of $P_Y(Y)$ on the output space requires evaluations of the mapping function $f(x)$ and/or its inverse (depending on the form estimator used). Uncertainty Quantification can be augmented by the use of surrogate models [12]. A surrogate model learns an approximation to a function which, if it can be evaluated more rapidly than the original function, can be used to facilitate approximation of $P_Y(Y)$. For PDE Uncertainty Quantification, if a distribution over inputs is known, then the surrogate model to be learned is map that calculates a PDE solution for a given input. For numerical PDE approximations, these functions can be computationally expensive to evaluate. Learning the surrogate model for real valued functions using known PDE solutions is a regression problem which is, following the discussion in Section 5.2, a form of supervised learning.

5.3.2 Polynomial Chaos and Support Vector Kernel approaches to Uncertainty Quantification

To compare to the ANN methods developed in this Section, Uncertainty Quantification based on Polynomial Chaos expansions is very briefly discussed.

Reflecting the popularity of Support Vector Machine techniques in Machine Learning, Polynomial Chaos approaches have been used extensively for PDE Uncertainty Quantification, for example see [356, 253, 210, 256, 254, 351]. Both Polynomial Chaos Expansions and Support Vector Regression fit a high dimensional regression surface to the known PDE solution values. These surfaces are frequently based on polynomial kernels functions [323, 81]. Polynomial Chaos models can be used in either an intrusive or non-intrusive mode. Non-intrusive solvers are independent of the PDE solver used to evaluate the PDE solution for a given input. Examples of non-intrusive Polynomial Chaos from the literature are presented in [177, 42, 104]. Non-intrusive surrogate models are particularly attractive from a practical point of view as it is far easier to use existing software than to implement new software. This is particularly true for complex cases such as PDE solvers. The ANN based surrogate models described in Section 5.3.3 are also non-intrusive in this sense. For Uncertainty Quantification, Polynomial Chaos methods often directly estimate the output distribution for a given input distribution. PCE based regression analysis is used to great effect by the Spectral Stochastic Finite Element Method [356]. §5 in [359] discusses a number of extensions to the basic SSFEM structure. Typically, though, varying the input distribution cannot be achieved without re-analysis. Probabilistic collocation can be used to mitigate some of these issues [240]. By contrast, the ANN based models explored in this Chapter are always independent of the input distribution used for the analysis.

In contrast with existing PCE and other kernel function methods [357, 356], ANNs do not require kernel basis functions to be pre-decided. Rather, ANNs adaptively learn basis functions to represent the data based on the more general form of the activation functions. This helps to displace the burden of feature design from the kernels to the design of the ANN [232]. In particular, because of the complexity of deep data representations, it is difficult to directly engineer kernels for these spaces. The two methods can, however, be combined [75]. Additionally, it is noted here that adaptive kernel methods have been used with PCE approaches [358]. The convergence rate of a Series Expansion Polynomial Chaos Stochastic PDE solution method is heavily dependent on the basis function selected, yet the optimal basis functions cannot be known a priori. For a PCE method, the number of terms required to reach a given convergence level increases with a factorial of the approximation order

[358]. If the expansion point is far from the point to be approximated or the shape of the basis is poorly suited to the function to be approximated, a large expansion order will be required. The factorial growth in the computational complexity of the approximation may render PCE methods intractable for certain problems. This is often the case for the far from mean response computations (crucial in Civil Engineering reliability problems [157]) for which the error in the tails of the distributed needs to be minimised.

5.3.3 PDE Uncertainty Quantification using ANNs

When applied to Uncertainty Quantification, supervised learning can be used as a surrogate model that estimates outputs for a given input. In addition to the challenges that apply generally to supervised learning (discussed in Section 5.2.2) probabilistic problems require high dimensional integration. A typical approach to integration using regression is to project the unknown function onto basis functions with known (or easily estimated) integral values. Integrals can then be easily approximated by factoring the basis function integrals appropriately by the regression coefficients. Directly integrating the functions represented by an ANN in this manner is not feasible for most problems. A general ANN model is the composition of transcendental functions of polynomials. Even given the regression weights, computing such an integral directly is very difficult. These models typically lack bijectivity which creates further issues. To avoid these problems, the function represented by an ANN can be integrated simply by Monte Carlo simulation. In this Chapter, the integration of trained surrogate models by Monte Carlo integration will be referred to as ANN-MCS and RNN-MCS for feedforward and RNN surrogate network architectures respectively.

The application of ANNs to PDE Uncertainty Quantification problems has been proposed previously, for example in §4.5.6 of [359], but has seen little traction. The historical issues with training ANNs (discussed in Section 5.2.2) are, as in Machine Learning more generally, the likely cause for the lack of progress made using ANNs for Uncertainty Quantification. By applying modern and deep learning methods, ANNs can be applied to boundary and initial value PDE problems, as demonstrated by the numerical experiments in Section 5.4. These numerical experiments also demonstrate that traditional Multilayer Perceptron networks [173] are less effective than modern deep rectified

networks as PDE surrogate models. This Section briefly describes basic use patterns for using feedforward and recurrent architectures with ANN surrogate models for discretised boundary value and initial value PDE problems.

Feedforward ANNs can be applied to both boundary and initial value problems. Consider a problem with a fixed geometry discretisation such as a static finite element mesh. For boundary value problems, the inputs to the feedforward network are taken to be all variable fields other than the solution field. These may include material coefficient fields, source terms, boundary conditions and others. The dimensionality of the ANN input will depend on the discrete representation of the variable fields used. The ANN is trained using the approximate numerical solutions of the PDE for a number of input values. The output dimension will be the size of the solution field discretisation. After training, the ANN will (hopefully) be able to predict the value of the solution field for a given input set. One method to perform Uncertainty Quantification for a given input distribution is to perform Monte Carlo simulation, replacing the PDE solver with the surrogate model. As the ANN surrogate is able to rapidly calculate values of the output field, Monte Carlo estimates for a large number of simulations can be run quickly. For initial value problems, feedforward networks can be used to predict the output of the PDE solver equation for a given input.

Recurrent Neural Networks can be used for initial value problem time series solution prediction. An RNN can be fitted to the sequence of solutions from a time stepping PDE solver, forming a surrogate model with time dependencies. This surrogate model can than be used for Uncertainty Quantification by applying Monte Carlo integration as in the feedforward case described previously. Importantly, although not fully explored in this Chapter, RNNs with memory can learn non-smooth functions in time with long range dependencies [23, 153]. This is in contrast to Series Expansion based methods [52]. Potentially RNNs could be used for time series prediction for physical simulations with sudden jumps in the solution function, such as collision simulations [89] or bifurcation problems, however training such networks a network is likely a challenging problem. RNNs with LSTM units are successfully used to build a surrogate model for an probabilistic heat flow equation finite element simulation problem in Section 5.4.4.

Although it can be empirically demonstrated that ANNs can be used as ef-

fective surrogate models for PDE Uncertainty Quantification (the main contribution of this Chapter), there are several limitations. These limitations affect ANN design across all ANN application areas. The choice of architecture cannot be known a priori. Further, the choice of training regime is also an unknown input to the ANN model design. Methods to determine effective architectures is an area of current research [151]. The ANNs used in this Chapter were found by experiment. ANN architecture design is a challenging problem, but because of the widespread adoption of ANNs across many fields, it is hoped that these issues will be resolved, allowing for the methods presented in this Chapter to be applied with greater ease in the future. In particular, Bayesian model design methods present a promising direction for future advances in this area.

5.3.4 Understanding the time complexity of supervised learning surrogate models for Uncertainty Quantification

This Section discusses the time complexity of regression based Uncertainty Quantification compared to direct Monte Carlo Simulation. It is necessary to ask whether the extra effort of learning a regression model is worth it when this computational time could be used to simply run a number of additional Monte Carlo simulations. In particular, many problems are run only once or only for a small number of input parameters. In these cases it is unlikely that the time spent on designing and training a surrogate model by regression will be worthwhile. By contrast, if a model is re-run repeatedly, then the time taken to train a surrogate model will become worthwhile if a number of conditions are met. This Section gives a rough overview of the computational complexity of applying regression models (in particular ANNs for ANN-MCS) to the task of learning the solution map of a differential equation for Uncertainty Quantification versus just running additional Monte Carlo simulations. This discussion is useful in that it suggests what conditions must be met in order for regression analysis to be worthwhile.

In the typical Uncertainty Quantification problem for probabilistic PDEs, a known distribution of inputs is fixed and a deterministic PDE solver (for the equation of interest) is available. The goal of Uncertainty Quantification is then to estimate the probability density of either the space of PDE outputs or expectations of Quantities of Interest (that are functions of this space of

outputs). One avenue for the application of supervised learning to Uncertainty Quantification using ANNs is as a method to generalise from the solution of one input problem to another, building up a surrogate model of the PDE solver mapping to avoid the wasteful repeated calculations that are present in sampling based techniques. Effective generalisation is the key to reducing the computational effort required for PDE Uncertainty Quantification using supervised learning.

Evaluation of an input using a deterministic PDE solver is typically computationally intensive. Sampling based methods, such as [320, 55], for estimating the probability density of the outputs will typically repeatedly sample from high density parts of the input space. This is wasteful as the same (or very similar) inputs to the PDE solver will be sampled and evaluated repeatedly. In the case of a finite and discrete input space with only a few possible inputs, it is obvious that the results of the evaluation of the mapping from inputs to outputs should be cached, allowing for the solution of a repeated input problem to be looked up quickly rather than recalculated. Then, sampling based methods for estimating the output probability density can proceed using only a single evaluation of the costly solver for each possible input, followed by a series of much cheaper lookups. Consider the case of MCS analysis with a finite input space of size m, a time to generate an input sample of c and a function evaluation time of k. Then, as MCS converges like $\sqrt{n}$ in n simulations (see [57]), the total cost for the naïve MCS to some fixed tolerance is $\mathcal{O}((k+c)\sqrt{n})$. On the other hand, if each storage takes time s and each lookup takes time l, the cached version of discrete MCS can be evaluated in time $\mathcal{O}(m(s+k+c)+(l+c)\sqrt{n})$. For the cached discrete MCS to be a faster algorithm, the following inequality must be satisfied:

$$\mathcal{O}(m(k+s+c)+(l+c)\sqrt{n}) < \mathcal{O}((k+c)\sqrt{n}) \tag{5.35}$$

Reorganising equation (5.35), for the case that the discrete cached MCS runs

faster than the original MCS:

$$\begin{aligned}
\mathcal{O}(m(k+s+c)+(l+c)\sqrt{n}) &< \mathcal{O}((k+c)\sqrt{n}) \\
\mathcal{O}(m(k+s+c))+\mathcal{O}((l+c)\sqrt{n}) &< \mathcal{O}((k+c)\sqrt{n}) \\
\mathcal{O}(m(k+s+c)) &< \mathcal{O}((k-l)\sqrt{n}) \\
\mathcal{O}(m) &< \mathcal{O}\left(\frac{(k-l)}{(k+s+c)}\sqrt{n}\right)
\end{aligned}$$

assuming further that $k \gg l$, $k \gg s$ and $k \gg c$ so that $\frac{(k-l)}{(k+s+c)} \approx 1$, the approximate inequality of interest is:

$$\mathcal{O}(m) < \mathcal{O}(\sqrt{n}) \tag{5.36}$$

Note that using the formal definition of big-$\mathcal{O}$ notation described in §3 of [80] this inequality could have been arrived at immediately but at the cost of some clarity. For $l < k$, in the case of $m = 0$ (the input is a single point), $\mathcal{O}(l\sqrt{n}) < \mathcal{O}(k\sqrt{n})$ is of course always satisfied. Equation (5.36) expands on this and says that the cached discrete MCS algorithm is the better choice for small m only, after which the inequality shown will fail to hold. This is also clearly visible from a simple graph of $f(x) = x - \sqrt{x}$ which shows x will dominate $\sqrt{x}$ for $x > 1$. Then, in the case of a continuous input space, m (in some discretised approximation of a continuous space) will become very large. As such, it would be more computationally efficient to perform the naïve MCS analysis without any precaching of all possible inputs than attempting to evaluate all possible inputs before probability density estimation on the output space. This is another manifestation of the so-called *curse of dimensionality* [32, 304], which renders certain algorithms intractable in high dimensional spaces.

The discussion on generalisation using supervised learning in the previous Section suggests, however, an alternative to computing all possible inputs for accelerating MCS density estimation. A small number of inputs can be generated from the known input distribution and the correct output calculated by the PDE solution function. An approximation to the output of the PDE solution function can be made using the ANN techniques for regression discussed in Section 5.2.2. Then, probability density estimation on the output space can be carried out by MCS with generated inputs and outputs estimated by an

ANN. In order for this generalisation to be worthwhile, the inequality (derived earlier) must hold:

$$\mathcal{O}(\hat{m}) < \mathcal{O}\left(\frac{(k-l)}{(k+\hat{s}+c)}\sqrt{n}\right) \tag{5.37}$$

where now, $\hat{m}$ represents the total number of training samples (such that $\hat{m} \in m$), c is the time to generate input vectors (for example random field simulations), $\hat{s}$ represents storage time as ANN training time per sample (discussed below) and l represents lookup time as ANN feedforward time per sample. There are some important changes in the ANN surrogate model case compared to the discrete cached MCS case. The condition that $\mathcal{O}(\hat{m}) < \mathcal{O}(\sqrt{n})$ must be true for to make the surrogate model computationally valuable is still valid. However the size of numerical constants (in particular the ANN training time, $\hat{s}$) are very significant and may render the algorithm not worthwhile even for small $\hat{m}$, especially if $k < \hat{s}$. Intuitively, this makes sense. If the function evaluation time, k, is very small many more MCS runs can be fit into the time that it would take to train an ANN to learn to represent the mapping (the time represented by $\hat{s}$).

In the discrete case the cached model had an identical MCS tolerance to the naïve direct MCS case. This is no longer true for the ANN surrogate case as the ANN function is only a hypothesis that is attempting to infer the structure of the true PDE solver function. In that case, the inaccuracy in the solution approximated by the ANN model can be represented as a part of the approximation storage time, $\hat{s} \geq s$. s represents the time taken to cache output sample data corresponding to a given input at maximum accuracy. $\hat{s}$ represents the actual time taken to store data (by training an ANN) to a given accuracy in a given function approximation. Equality could be possibly achieved ($\hat{s} = s$) only when the output sample for a given input can be stored at maximum accuracy. For example, in the discrete case s is the cost to store an $(input, output)$ pair. In this case, $\hat{s} = s$ as one simply needs to store the bits corresponding to the particular output. In the case that an ANN is used as a function approximator, the lookup (feedforward) time is fixed but it may take multiple rounds of training to converge the ANN to an acceptable tolerance for a given output by gradient descent, so $\hat{s} \geq s$. In this case equation (5.37) can be modified and the time complexity inequalities that must be satisfied

represented as:

$$\mathcal{O}(\hat{m}(k+s+c)) \leq \mathcal{O}(\hat{m}(k+\hat{s}+c)) < \mathcal{O}((k-l)\sqrt{n}) \tag{5.38}$$

Equation (5.38) expresses the intuitive result that the increased time cost required to store $\hat{s}$, rather than s, hurts the efficiency of the function approximation algorithm relative to direct MCS. A fast function approximation algorithm is required to make the ANN function approximation model computationally competitive.

The important part of this analysis is that in order to make regression competitive against direct Monte Carlo, several factors must be considered. First, the regression model must be sufficiently accurate. Second, there is a time cost in designing features for a regression surrogate model. The time spent on feature engineering and problem set up for a complicated regression method, such as SVM and Polynomial Chaos [47] based approaches using sparse polynomials [49], may perhaps be better put to use by running a Monte Carlo estimator for a long time in a "fire and forget" manner, running the solver in a simple loop. By using a more general adaptive method, specifically ANNs, the feature engineering time can be reduced. However, the feature design issue is displaced from the function selection in the SVM or Polynomial Chaos case to the design of the ANN architecture. Finding useful ANN architectures is still a type of feature engineering problem. For example, the numerical analyses in Section 5.4 demonstrate that certain architectures perform well while others are less accurate. In particular for the heat equation-based analyses shown, rectifier units provide a significant ANN convergence performance boost when compared to more traditional Mulitlayer Perception networks. However, the design of efficient ANN architectures is still difficult. There is the potential that an ANN could be trained for many problems across Uncertainty Quantification and applied to new problems by transfer learning [267, 376]. This is will be a challenge for future research. Bayesian model selection for ANNs is also a promising direction from which future developments in this area may appear [275].

The regression model must also be able to be trained efficiently. If the regression model training is too slow, then the computational resources spent on the regression would be better spent on increasing the number of Monte Carlo analyses. Training deep ANNs can be problematic, but advances like rectifying

activation functions has helped to unlock the potential of deep function representations. Training the Multilayer Perceptron network models with more than two or three layers became almost impossible [173]. It was not feasible to apply these models to anything but simple problems [232]. Wide, shallow networks were favoured, discarding the advantages of deep data representations, as these were the only networks that could be trained efficiently. Modern ANN techniques have mostly been able to address the vanishing (or exploding) gradient issue that was the primary cause of difficulty when training Multilayer Perceptron networks, as discussed in Section 5.2.2. In general, ANNs require large amounts of training time and data. Increases in computational power have made the application of ANNs to PDE Uncertainty Quantification problems feasible in part by allowing for the networks of a reasonable size to be trained sufficiently quickly. The design of an appropriate training schedule is challenging and currently requires a good understanding of the factors causing ANNs to fail to converge. While existing optimisers do partially adjust learning rates adaptively (see discussion in Section 5.2.2.4), further improvements will be needed in this area to allow for broad uptake of ANN methods in practice. Due to the widespread adoption of ANNs in a large number of areas, research into these issues can be expected to continue and accelerate.

Importantly, the time complexity analysis above has been taken largely given from the perspective of a single use analysis. For a model that is to be used repeatedly, ANNs have the advantage that, once trained, they can be used multiple times for different input distributions. For example, if the input distribution is changed, direct Monte Carlo analysis would have to be restarted from the beginning. An ANN trained for one input distribution could be reused with no (or perhaps minimal) retraining because the training is independent of the Uncertainty Quantification input distribution. Consider the example of weather forecasting. New random data is collected frequently from weather stations, the numerical models are very large and analyses are repeated many times. A trained ANN could be used to facilitate rapid Uncertainty Quantification given the latest data and input value distributions. When models are to be used repeatedly with different inputs, the relative time cost of the data caching (that is, training) for generalisation is reduced. Further, if the inference accuracy is high, then it would be increasingly worthwhile to train an ANN for PDE Uncertainty Quantification. Variable input distributions are considered in the literature (for example [84]), however for large scale

PDE problems these methods add additional computational difficulties beyond those already faced for a fixed input distribution. By utilising ANNs for generalisation, it will be easier to consider variability in the input distribution itself for computationally expensive PDE problems.

The overall conclusion of this analysis is that for single use PDE Uncertainty Quantification problems that can be solved very efficiently, ANN regression is unlikely to be competitive compared to simply running a large number of MCS analyses. However, there are several important cases where training ANN surrogate models is worthwhile. Complex PDE problems that require significant computational overheads to solve are likely to benefit from the proposed ANN method by caching the costly known solutions. For models that are used repeatedly (such as for weather forecasting), the time taken to train the ANN will likely be made up for by avoiding running costly PDE solvers. Additionally, for parametric studies, sensitivity analysis and Uncertainty Quantification (all models with repeated evaluations of a model) it is likely to be worth the computational effort to train an ANN surrogate. The numerical experiments presented in Section 5.4 demonstrate that the ANN augmented output probability distribution are more accurate than those computed by MCS and so ANN-MCS would also be worthwhile when the training data examples are very difficult to obtain.

5.3.5 Kernel Density Estimation

The output probability distribution can be estimated from point data by a variety of techniques. Histograms are the simplest method and are used for the third numerical experiment presented in this Chapter [128]. For the first two numerical experiments in this Chapter, as the focus is not on advanced density estimation techniques, a simple Kernel Density Estimator (KDE) is used for the numerical analyses in Section 5.4. While technically the choice of a kernel is a form of feature engineering, Kernel Density Estimators were found to work well for all of the test case problems considered without any unreasonable efforts. Kernel Density Estimators have the useful feature that they have representations that can be integrated easily, although this was not used for the numerical analyses presented in this Chapter. Estimating Quantities of Interest for problems such as reliability estimation requires integration over the output space and as such KDE models can be used to facilitate QoI

analyses.

Following §7.2 in [224], §2.5 in [46] and [339, 330], given a set of N data points $(d_1, \cdots c_N)$, drawn independently from an unknown probability density, $f(d)$, the Kernel Density Estimator, $D(d)$, of an unknown probability density, f, is defined as:

$$f(d) \approx D(d) = \frac{1}{N} \sum_{i=1}^{N} K(d - d_i; h) \tag{5.39}$$

where h is a smoothing parameter called the bandwidth. The bandwidth must be selected as a part of the model fitting process and controls the bias-variance tradeoff of the model [224]. There are a variety of methods for selecting appropriate bandwidths, but these are not discussed in detail here. For simplicity, only fixed bandwidth density estimators are used for the analyses in Section 5.4.

The function $K(u)$ is called a *kernel function* and is taken to be a a non-negative function with expectation zero and unit integral. There are a variety of choices of kernel functions available [10]. However, in this Chapter attention will be restricted to the Gaussian kernel in the form used by the Scikit-learn Python library [293]:

$$K(u; h) \propto \exp\left(-\frac{u^2}{2h^2}\right) \tag{5.40}$$

5.4 Numerical analyses

This Section demonstrates how deep ANNs can be applied to problems in Uncertainty Quantification for PDE problems. Three test case problems in order of increasing complexity are considered, each based on the heat equation. First, the steady state boundary value Poisson problem is considered. In this experiment, the task is to estimate the output distribution given random, spatially correlated diffusion and source fields. Kernel Density Estimates produced by a small number of direct Monte Carlo Simulation analyses are compared to those by ANN-MCS (that is ANN regression augmented MCS). The second and third numerical experiments are nonlinear initial value problems, reintroducing the time dependent part of the heat equation. Further, the diffusion coefficient is taken to be a nonlinear function of the current temperature. For these analyses, the initial value field is uncertain and modelled

as a spatially correlated random field. The second numerical experiment is similar to the first, except the goal is to estimate the uncertain solution field after a set number of time steps. The Kernel Density Estimates results using data from direct Monte Carlo Simulation and ANN-MCS are compared. In the final experiment, the goal is to estimate the entire sequence of values for the solution field across all time steps. For this purpose, the performance of a Recurrent Neural Network for RNN-MCS is compared to direct Monte Carlo Simulation. For each numerical experiment, the results are verified against a direct Monte Carlo analysis with a large number of simulations.

In all cases, the random fields and boundary conditions have been selected to ensure that a steady state distribution is reached. The Heat equation PDEs analysed will converge to a steady state in all cases as fixed Dirichlet boundary conditions and a fixed (random) source field are used. Further, the ANN model only reads output results from the PDE solver. As such, the ANN-MCS technique will not influence the steady-state distribution of the numerical FEM solver. There are, however, no guarantees that an ill-fitting surrogate model will reach any sensible steady state. In the limit that an ANN surrogate model is perfectly accurate, it will output the same steady state condition as the PDE solver.

5.4.1 Specification details common to each numerical experiment

5.4.1.1 Equations and numerical PDE solver specifications

Each of the numerical experiments in this Section are based on solutions of the heat equation. Following the notation in [246]:

$$\frac{\partial u(\omega)}{\partial t} = \nabla \cdot (g(\omega)q(u)\nabla u(\omega)) + f(\omega) \qquad \text{in } \Omega \times [0,T] \tag{5.41}$$

$$u(\omega) = u_D = 0 \qquad \text{on } \partial\Omega \times [0,T] \tag{5.42}$$

$$u(\omega) = u_0(\omega) \qquad \text{at } t = 0 \tag{5.43}$$

where ω are points in the spatial domain Ω, $\partial\Omega$ is the boundary of this domain, the time is given by t taken to run from 0 to T. For all numerical analyses, the spatial domain was taken to be a unit square in $\mathbb{R}^2$. For all three numerical problems considered, the boundary condition is taken to be $u_D = 0$ at all

times. The field $f(\omega)$ is termed the source field. The field $g(\omega)$ is termed the material coefficient field. The field $u_0(\omega)$ is termed initial condition field. Finally, the function $q(u)$ is termed the constitutive model function.

All numerical analyses of equation (5.41) were carried out using the FEniCS finite element library [9]. In particular, the solver code in §1 of [246] was used with only minor modifications. For each of the numerical experiments in this Section, the same finite element mesh geometry is used. This mesh consisted of 128 elements P1 elements (see [246]) with a total of 81 nodes arranged in a 9 by 9 vertex grid. The discretised problem geometry is shown in Figure 5.5.

For all numerical experiments, the input fields $f(\omega)$, $g(\omega)$ and $u_0(\omega)$ are assumed to be uncertain. Each of these input fields was taken to be distributed independently from one another. Each uncertain input field is modelled probabilistically as a spatially autocorrelated random field with exponentially decaying correlation function (see [5] for more details):

$$\rho(\omega_1, \omega_2) = \exp\left(-\frac{\|\omega_1 - \omega_2\|^2}{\gamma}\right) \tag{5.44}$$

where $\|\omega_1 - \omega_2\|^2$ is the square of the distance between ω_1 and ω_2 and γ is the correlation length parameter. For all random fields analysed, the correlation length scale was set to $\gamma = 0.25$. This represents a intermediate amount of correlation and will force simulated random fields to be undulating over the scale of the problem domain. The source and initial value fields were taken to be zero mean Gaussian random fields with unit standard deviation and covariance structure induced by equation (5.44). To ensure positivity of the heat equation diffusivity term $(g(\omega)q(u))$, the material coefficient field is set to be a lognormal random field with zero mean, standard deviation 0.25 and covariance structure induced by equation (5.44) applied to the underlying Gaussian field. In summary, the uncertain input distribution was taken to

be:

$$
\begin{aligned}
P(X(\omega)) &= P(f(\omega)) \times P(g(\omega)) \times P(u_0(\omega)) \\
P(f(\omega)) &= \mathcal{N}\left(\vec{0}, \rho(\omega_1, \omega_2)\right) \\
P(u_0(\omega)) &= \mathcal{N}\left(\vec{0}, \rho(\omega_1, \omega_2)\right) \\
P(g(\omega)) &= \exp\left(\mathcal{N}\left(\vec{0}, 0.25^2 \times \rho(\omega_1, \omega_2)\right)\right)
\end{aligned}
$$

where $\mathcal{N}(\vec{a}, \sigma^2\rho(\omega_1, \omega_2))$ denotes a normal distribution with mean vector with all values equal to a, standard deviation σ and correlation as defined by equation (5.44).

Random fields on the spatial domain were all simulated using the same underlying Gaussian random field generator. The Gaussian random fields were generated by taking the eigenvalue decomposition (KL Expansion) of the correlation structure discretised by the finite element mesh. Each is generated by a transformation, L, of a vector of Independent Identically Distributed (IID) standard normal samples, α such that a correlated Gaussian random field is given by $\beta = L\alpha$. Lognormal Gaussian random fields, γ, are simulated by applying the elementwise transform $\gamma = \exp(\beta)$ to an underlying Gaussian random field simulation. The full method is detailed in [158, 118]. All random field simulations were generated using the matrix operations in the Python library NumPy [380].

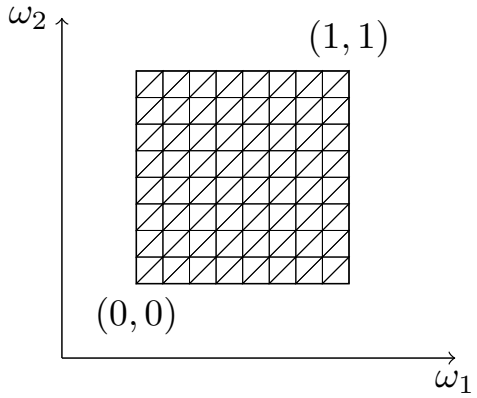


Figure 5.5: Finite Element Mesh adopted for all analyses on domain $\Omega = [0,1]^2 \in \mathbb{R}^2$ consisting of 81 nodes and 128 evenly equally sized triangular P1 (see [246]) elements.

5.4.1.2 Analysis methodology specification

For each of the numerical experiments in this Section, roughly the same methodology was used. To test the performance of the proposed ANN based regression methods, three analyses were completed. First, a solution estimate was generated for verification using 1×10^5 direct Monte Carlo analyses. A KDE of the output distribution at each node, ω, in the finite element mesh was constructed using the results of this verification analysis. This estimated density will be referred to as the *verification KDE*, $D_v(\omega;\xi)$. To compare the performance of direct MCS and ANN-MCS, the error of the KDEs calculated by these methods compared to the verification KDE was approximated using the method described below in Section 5.4.1.3.

For the first two numerical test cases, the performance of MCS and ANN-MCS was compared by the KDE error at a point within the spatial domain. The methods tested are direct Monte Carlo analysis and a ANN-MCS for a number of ANN architectures. Each method was tested using a set number of known values of the solver function. Specifically, the number of known values of the function to be evaluated was restricted to one of elements of the set $\{1000, 2000, 3000, 4000, 5000\}$. For MCS analyses, known function values were found by sampling from the input distribution and then running the FEM solver. For the ANN-MCS analyses, the samples were generated by scaled input sampling by the method described in Section 5.4.1.4 below. To further test the different ANN architectures, when testing the ANNs with different training set sizes, the network weights were reinitialised. As described in Section 5.2.2.4, Stochastic Gradient Descent optimisation finds local minima. By restarting the ANN analysis randomly each time the same network architecture type was analysed, the effect of the weight initialisation starting point was minimised. The different architectures trialled could be compared more effectively by removing the confounding effect of the starting state of the weights.

For the third numerical experiment, the RNN-MCS study, a single architecture was tested using 1000 data points for MCS and RNN-MCS. Accuracy was assessed by comparison to a verification analysis of 1×10^5 direct Monte Carlo runs. The RNN-MCS integration was performed using 1×10^4 integration trajectory simulations. Further, the RNN was restricted to only predict a solution field value at a single point within the output space, rather than

the entire solution field (which was modelled for the ANN-MCS tests).

5.4.1.3 KDE specification

For the different numerical experiments presented, a number of Kernel Density Estimates are compared. The value of a probability density estimate for a given value using a Kernel Density Estimator is denoted by:

$$P_\omega(\xi) = D_{ID}(\omega, \xi) \tag{5.45}$$

where ID is an identification label, ω refers to the spatial coordinate in the finite element mesh and ξ refers to the random variable value assigned a probability by $D_{ID}(\omega, \xi)$. For the first two numerical experiments, errors were assessed as follows. Given an approximately correct verification value KDE $D_v(\omega, \xi)$, the error of another KDE estimate, D_a, for a location ω is given by the sum of squared errors:

$$E(\omega; D_a) = \|D(\omega, \xi) - D_{check}(\omega, \xi)\|^2 = \int_{\mathbb{R}} (D(\omega, \xi) - D_{check}(\omega, \xi))^2 d\xi \tag{5.46}$$

For all analyses in the first two numerical tests, this value is approximated by taking the predicted KDE values at 2000 points evenly spaced over points within a fixed distance from $\xi = 0$. Note that KDE's for the first two numerical experiments are rescaled to the correct probability density values by taking into account the spacing between the 2000 approximation points. Another type of error estimator is used for the time series prediction in the third numerical test case and is described in Section 5.4.4.

The Scikit Kernel Density Estimator [293] was for all KDEs. For all analyses, fixed bandwidths were used. The value of these bandwidths are given in each of the numerical experiments. These bandwidths were tuned manually. A more careful analysis of appropriate bandwidths would be required in certain instances.

5.4.1.4 Artificial Neural Network Details

All ANNs were generated using the Keras software library [76] to interface with Theano [371]. Weight tolerances were set to 64 bit precision (Python 'Float64'

[380, 204]) and the learning accuracy tolerance of was set to 1×10^{-8}. All ANN training was carried out using the Adam optimiser (see [212] and Section 5.2.2.4) with a mini-batch size of 10. Learning rates were set differently for the different numerical experiments presented and are described in the Sections 5.4.2, 5.4.3 and 5.4.4. With reference to Section 5.2.1, a mean squared error loss function of the form:

$$J(\theta) = \frac{1}{2}\|output - target\|^2 = \frac{1}{2}\|h(x) - y\|^2 \tag{5.47}$$

was used for ANN training in all cases tested.

As ANNs are poor at extrapolation but effective for interpolation tasks [368], the input samples were generated by spread out samples from the input distribution. To spread out the ANN samples, before applying the correlation transformations L to the IID standard normal vector, a scaling factor of λ was applied to α such that for the ANN analysis:

$$\beta = L(\lambda\alpha) \tag{5.48}$$

Different scaling factors were used for different analyses and are described in the relevant Sections.

5.4.1.5 ANN and RNN Architecture specification

For the numerical analyses, a number of different ANN architectures were tested. To simplify the description of these architectures, they are described using the framework in this Section. Each ANN architecture is given as a set of layer connections and activations. Each architecture is specified by a set of size $N + 1$ of the following form:

$$Name := \{X_0, C(X_1, A_1), C(X_2, A_2), \cdots, C(X_N, A_N)\}$$

where $Name$ is an identifier, X_0 is the input dimension and $C(X_i, A_i)$ is a layer connection that outputs with X_i units and then applies an activation function A_i. If a value $C(X, A)$ is repeated p times, the shorthand notation $C(X, A)^p$ is used.

The following layer connections are considered:

- $D(X, A)$: Fully connected layer - see Section 5.2.2.1
- $M(X, A)$: Long Short Term Memory (LSTM) connection - see Section 5.2.2.3

The following activation functions (choices for A) were considered:

- L: Linear - equation (5.9)
- S: Sigmoid - equation (5.10)
- T: Hyperbolic Tangent - equation (5.11)
- R: Rectified Linear Unit (ReLU) - equation (5.12)
- E: Exponential Linear Unit (ELU) - equation (5.13)

5.4.2 Steady state linear Poisson Equation

The first test case considered is the simplest. A two dimensional steady state heat conduction boundary value problem is analysed using the Finite Element Method. Random forcing and diffusion coefficients are modelled by random fields. The goal of the analysis is to generate a Kernel Density Estimate of the output field at a particular point (the centre) of the spatial domain. Two probabilistic analysis methods are compared. First, a direct Monte Carlo approach is used to estimate the distribution of the output space. Next, a deep ANN is used as a surrogate model for ANN-MCS using the same FEM solver. Several different ANN architectures are considered. The Kernel Density Estimates of the output field at the central point calculated by both methods are compared.

5.4.2.1 Problem definition

With reference to Section 5.4.1 and equation (5.41), the equation analysed for this first numerical experiment is the Poisson Equation:

$$-\nabla \cdot (g(\omega)q(\omega)\nabla u(\omega)) = f(\omega) \tag{5.49}$$

As there is no time dependence, the initial value of the field u_0 is irrelevant for this problem. For this linear problem the constitutive field is constant, that is, $q(u) = 1$.

The goal of this analysis is to estimate the probability density of the solution field to equation (5.49). In particular, the aim is to train an ANN to predict the value of the solution to equation (5.49). These predictions are then to be used to provide improved Kernel Density Estimates (compared to Monte Carlo Simulation) of the probability density of $u(\omega)$. For each of the ANNs tested, to allow for comparison between each, training is restricted to 1000 epochs. A fixed KDE bandwidth of 0.0008 was used for the verification KDE as well as all ANN-MCS KDE. A fixed bandwidth of 0.0012 was used for all test MCS analyses as this higher bandwidth reduced the MCS error relative to the verification analysis. These values were selected empirically based on the criteria described in [46] to be neither too small (insufficient smoothness of output distribution) nor too large (the output density is exactly the kernel density). A more detailed and complicated analysis could be performed using the likelihood of the kernel parameter using a Gaussian Process formulation. For the demonstrations presented in this Section, the empirical approach is sufficient.

5.4.2.2 ANN details and architectures

A fixed learning rate of $\eta = 0.001$ was used for all analyses. A training set sampler scaling factor of $\lambda = 2.0$ (see Section 5.4.1.4) was used for all ANN analyses. With reference to the definitions in Section 5.4.1.5, the following ANN architectures were considered:

- *Deep Rectified* $:= \{I, D(I,S), D(I,R), D(O,S), D(O,R), D(2{\times}I,R)^2, D(O,L)\}$
- *Wide Sigmoid* $:= \{I, D(I,S), D(2000,S), D(O,L)\}$
- *Multilayer Perceptron (MLP)* $:= \{I, D(I,S), D(2{\times}I,S)^3, D(O,L)\}$

where $I = 81{\times}2 = 162$ is the common input dimension, $O = 81$ is the common output dimension to the ANN. Note that the input dimension, I, of each ANN is the size of the coefficient vector field, $g(\omega)$, plus the source vector field, $f(\omega)$. The output dimension of each ANN is the dimension of the solution field, $u(\omega)$. The *Deep Rectified* network consists of several layers of ReLU and sigmoid activations. The *Multilayer Perceptron* network is a traditional architecture consisting of only sigmoid activations. The *Wide Sigmoid* network is a wide, single layer Perceptron network.

5.4.2.3 Numerical Results

The analysis was carried out following the description given in Section 5.4.1.2. Fixed bandwidth KDE estimates were calculated for direct Monte Carlo Simulation and for each of the ANN architectures described in Section 5.4.2.2. Performance measures of each of the methods tested for estimating the probability density of $u(\omega = (0.5, 0.5))$ are presented. Note that $\omega = (0.5, 0.5)$ is the centre of the spatial domain.

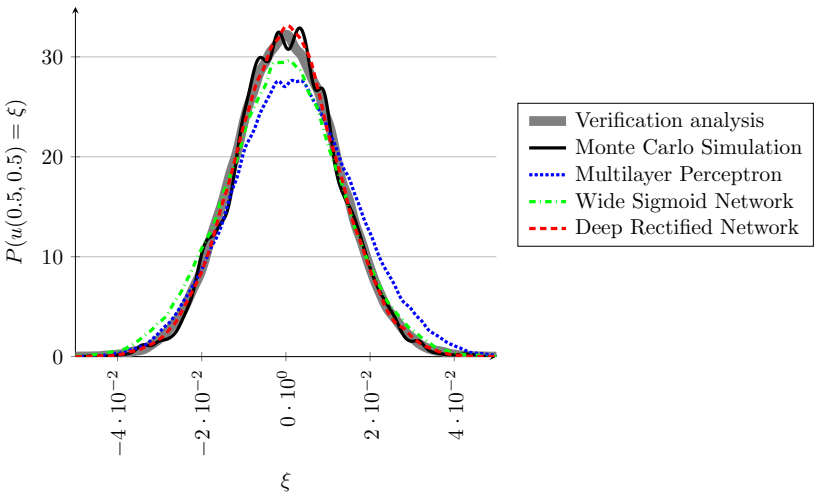
The shape of the KDE estimators and their errors using a training set size of 5000 known values is summarised in Figure 5.6. Figure 5.6a plots the Kernel Density Estimates for the verification analysis, $D_v(\omega = 0.5, 0.5, \xi)$, and for each of the methods tested. Figure 5.6b plots the error (as the difference) between the KDEs in 5.6a and the verification density. Table 5.1 presents the error $E(D_i, \omega = 0.5, 0.5)$ for each KDE D_i compared to the verification density, $D_v(\omega = 0.5, 0.5, \xi)$, for direct Monte Carlo Simulation and for each of the ANN architectures described in Section 5.4.2.2. Figure 5.7 plots the estimator error versus number of input values using the data from Table 5.1 for each of the methods tested.

Method	Type	KDE Error per number of training examples				
		Number of training examples				
		1000	2000	3000	4000	5000
MCS	N.A.	0.81446	0.87507	0.81473	0.57427	0.44319
ANN	Rectified	1.13293	0.62516	0.53372	0.14418	0.09717
ANN	Wide Sigmoid	0.55119	0.67793	0.63492	0.51531	0.80421
ANN	MLP	4.55736	3.58707	3.75562	2.21758	2.38577

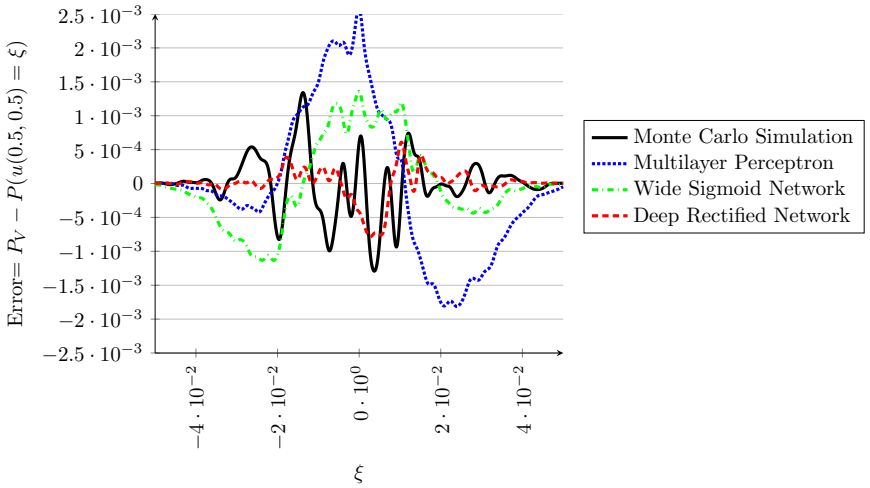
Table 5.1: Steady State Poisson Equation Kernel Density Estimator (KDE) error for $u(0.5, 0.5)$ (the central point in the spatial domain) calculated for each numerical method tested. Errors are calculated for the specified number of training examples. Error is estimated as the sum of squared errors relative to the 1×10^5 verification KDE MCS analyses as per the description in Section 5.4.1.3. The method column refers to either direct Monte Carlo Simulation (MCS) or Artificial Neural Network (ANN) regression for ANN-MCS. Type N.A. denotes 'not applicable'. The type column for ANN methods refer to the ANN architectures in Section 5.4.2.2.

5.4.2.4 Discussion

The numerical results for the linear Poisson problem problem directly demonstrated that feedforward ANNs can be used as surrogate models for FEM



(a) Kernel Density Estimates of $u(0.5, 0.5)$



(b) Errors of Kernel Density Estimates of $u(0.5, 0.5)$ compared to verification analysis.

Figure 5.6: Steady State Poisson Equation Kernel Density Estimates and errors. Figure 5.6a shows, for each method tested, the KDEs of $u(0.5, 0.5)$ which is the solution to probabilistic Poisson Equation described in Section 5.4.2.1 at the centre of the spatial domain shown in Figure 5.5. Figure 5.6b shows the error of each of these KDE estimates compared to the verification analysis KDE (from 1×10^5 direct Monte Carlo Simulations).

boundary value PDE solutions. If the right type of ANN architecture is used, these surrogate models can generalise well and estimate probabilities for Uncertainty Quantification more accurately than direct Monte Carlo Simulation. The ANN architectures used were found by trial and error, but the presented results should aid future design efforts. Further, the numerical results demonstrated a number of anticipated results. For the sigmoid only networks, the

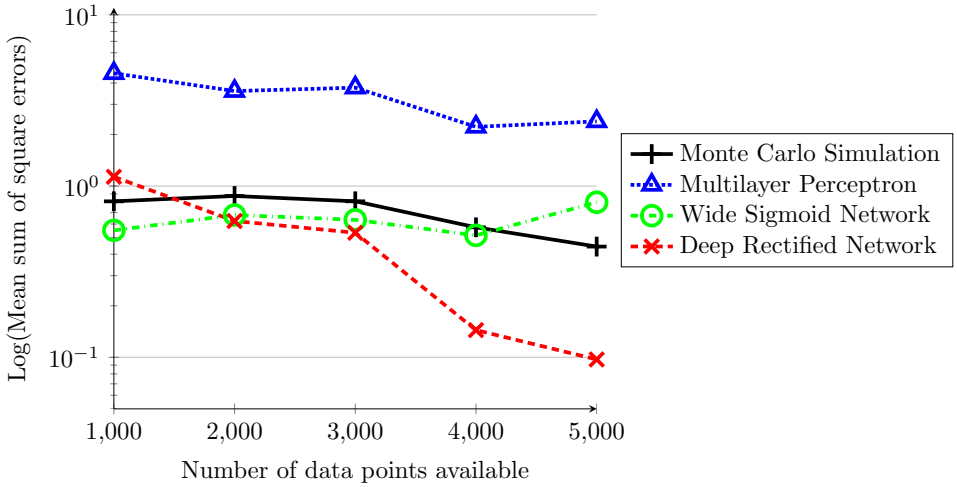


Figure 5.7: Comparison of errors for MCS and different ANN surrogate model architectures for the Steady State Poisson Equation analysed. Monte Carlo Simulation analyses were calculated from Kernel Density Estimates using the the number of analyses shown on the horizontal axis. The three ANN models tested were fit using training set of size shown on the horizontal axis. A Kernel Density Estimate was then built from 1×10^5 simulations from the surrogate model. Errors are calculated as the mean sum of squared differences from between the Kernel Density Estimate using the different function approximation techniques and the verification analysis KDE (from 1×10^5 direct MCS analyses). Note that the errors were calculated with a fixed width Gaussian kernel for all analyses, as such the error shown should be considered relative to one another and not as absolute error indicators (which are kernel bandwidth dependent).

wide and shallow network outperformed the deep Multilayer Perceptron network. This was expected based on the discussion in Section 5.2.2. Additionally, it was shown that deep networks with rectifier units can be used as accurate function approximation surrogate models.

Figure 5.6a demonstrates that all of the networks tested converged roughly to the output probability density for $u(0.5, 0.5)$ predicted by the verification analysis (direct MCS with 1×10^5 simulations). The error analyses in Figures 5.6b and 5.7, as well as Table 5.1, indicate that that the deep network with mixed ReLU and sigmoid activations converged the most quickly to the correct output probability density, converging more quickly than either of the sigmoid only networks. The shallow, wide sigmoid network performed better than the deep sigmoid network. The deep sigmoid only network did not completely finish converging to the correct distribution. This is likely due to the fixed training length (1000 epochs) that was used. With additional training it is possible that the deep sigmoid network would have converged more closely

to the correct distribution. The reasons for the superior performance of the network with rectifier units follows from the discussion in Section 5.2.2.5.

The nonlinearity in the convergence trends visible in Figure 5.7 for each network tested are due to the random reinitialisation of each networks weights for each training set size considered. Further, the training data points are randomly reselected for every analysis. This methodology is as described in Section 5.4.1.2. As SGD is a local optimisation algorithm, the starting network weights will impact on the ANN weights at the end of training. The deep ReLU sigmoid mixture network displayed a more consistent improvement with increasing training data than the other networks tested. In order to test this convergence, it was found to be useful to use the results of MCS sampling in order to verify the shape of the ANN predicted output distribution. For a realistic analysis, the solution (in the form of the verification analysis in this case) would not be available. This means that errors of the type shown in Figures 5.6b and 5.7, as well as Table 5.1, would not be available as a means for assessing the quality of the ANN approximation. Of course, standard supervised learning validation techniques such as 'leave one out cross validation' (for details see [269]) could be used to assess the performance of the ANN. However, as even a small number of MCS samples can be used to easily approximate the rough shape of the output distribution, these MCS samples could be used to check the convergence of the estimated output probability distribution with less computational overhead than methods like leave one out cross validation. For example, with reference to Table 5.1, the ANN predicted output distribution should converge in the direction predicted by an increasing number of direct MCS samples. It is important to note that, in this context, ANNs learn the function from inputs to outputs and is not dependent on the input distribution selected for the Uncertainty Quantification problem. That is, the training examples need not follow the input distribution to be analysed. This is in contrast with direct MCS which estimates the output distribution for a given input set. As such, care should be taken when using MCS density estimation as an ANN validation technique for Uncertainty Quantification problems with variable input distribution or when the training set is not composed of independent samples drawn from the input distribution.

The overall result of the analysis was that by using an appropriate ANN, deep networks can be used as very effective surrogate models for steady state

boundary value PDE Uncertainty Quantification problems. By making the right choices regarding training techniques and activation functions, deep architectures can be used effectively. The most challenging aspect of the design of this numerical experiment was the selection of an appropriate deep architecture. The results are not shown, but deep networks consisting only of ReLU units performed very poorly. The combination of units with nonlinear and linear gradients likely simultaneously improved the ability of the network to calculate the required functions while preventing vanishing or exploding gradients. Further, the deep ReLU sigmoid mixture network features a bottleneck after a few layers. This bottleneck drops the dimensionality of the network by mapping from a subset of $\mathbb{R}^{162}$ to a subset of $\mathbb{R}^{81}$. The network then widens out again, to a subset of $\mathbb{R}^{324}$, before reaching the output space, $\mathbb{R}^{81}$. The bottleneck architecture was found to be very effective for this numerical experiment. This is likely because, when considering the structure of the problem being solved (essentially matrix inversion for different vectors and matrices), there are sparse connections between the input space features initially but dense interactions between the input components after matrix inversion. The wide part of the network allows for the network to find which interactions of input components are relevant. By using a bottleneck, the network is forced to pick a small number of dominant principle components in a way that is similar to dimensionality reducing auto encoders [184]. As the function to be learnt is the PDE solution output, it is sensible to set the bottleneck width (number of principle components) to be the dimensionality of the output field. Although the bottleneck architecture was successful for this numerical experiment, it was found primarily by trial and error. Improving techniques for finding useful ANN model architectures remains an open area of research [151].

5.4.3 ANN-MCS analysis of an initial value nonlinear heat equation problem

The second numerical problem extends the first by reintroducing the time dependency in heat equation, see equation (5.41). This is an initial value problem, rather than the boundary value problem presented in Section 5.4.2. The initial conditions for this problem are uncertain and modelled as a spatially autocorrelated random field. The values of the solution field, $u(\omega)$, at the final analysis time at the central point within the spatial domain are to be estimated. As a further modification, nonlinearity is introduced to the diffusion coefficients. The introduction of these nonlinearities makes ANN training more difficult than in the linear case.

5.4.3.1 Problem description

With reference to Section 5.4.1, the heat equation described in equation (5.41) was solved for the constitutive model:

$$q(u) = 1 + u^2 \tag{5.50}$$

For numerical analysis, time was taken to run from $t = 0$ to $t = 1$ in 50 discrete and evenly spaced intervals.

For this numerical experiment, the goal is to estimate the probability density of $u(\omega)$ at $t = 1$ given the probabilistic input random fields for the material coefficient, source and initial value fields given by $g(\omega)$, $f(\omega)$ and $u_0(\omega)$ respectively. The probability distributions for the input fields are given in Section 5.4.1. In particular, the aim is to train an ANN to predict the value of $u(\omega)$ at $t = 1.0$ given the input field values. These predictions are then to be used to provide improved Kernel Density Estimates (compared to Monte Carlo Simulation) of the probability density of $u(\omega)$ by ANN-MCS. A fixed KDE bandwidth of 0.0015 was used for all analyses. This value was fit empirically based on the observed data to be neither too small (rendering the output PDE very rough) or too large (output PDE hidden by kernel function). For further discussion regarding the selection of kernel parameters, see Section 5.3.5. Note that the estimated error for the method tested is dependent on the KDE bandwidth. As the (non-verification) MCS analyses use a

lower number of density estimation points to calculate the KDE compared to the ANN methods, the MCS KDE error is likely to be higher than if a higher bandwidth was used. The errors between the two ANN methods tested can be compared directly, but the MCS error should be considered as an indicative relative error.

5.4.3.2 ANN details and architecture

To train the ANNs for this numerical experiment successfully, a reducing learning schedule was used. The following 100 epoch schedule was adopted for all ANN analyses:

1. $\eta = 1{\times}10^{-3}$ for 20 epochs
2. $\eta = 1{\times}10^{-4}$ for 20 epochs
3. $\eta = 1{\times}10^{-5}$ for 20 epochs
4. $\eta = 1{\times}10^{-6}$ for 20 epochs
5. $\eta = 1{\times}10^{-7}$ for 20 epochs

This training schedule was found by trial and error to perform well. The reducing learning rate schedule is simplified form of simulated annealing. A training set sampler scaling factor of $\lambda = 1.1$ (see Section 5.4.1.4) was used for all ANN analyses.

With reference to the definitions in Section 5.4.1.5, the following ANN architectures were considered:

- *Deep ELU & Tanh* $= \{I, D(I,L), D(W,T), D(W,E), D(W,T), D(O,L)\}$
- *Multilayer Tanh* $= \{I, D(I,L), D(W,T), D(W,T), D(W,T), D(O,L)\}$

where $I = 81{\times}3 = 243$ is the common input dimension, $O = 81{\times}3 = 243$ is the common output dimension to the ANN and $W = 10{\times}O = 2430$. The input and output dimensions are the sum of the dimensions of the coefficient vector field, $g(\omega)$, the source vector field, $f(\omega)$ and the solution field, $u(\omega)$. The *Multilayer Tanh* (or ML Tanh) network is a traditional Multilayer Perceptron architecture with three tanh activation layers. The *Deep ELU & Tanh* network consists of two tanh activation layers with an ELU nonlinear rectifier layer in the middle of the network.

5.4.3.3 Numerical results

The analysis was carried out following the description given in Section 5.4.1.2. KDE estimates were calculated for direct Monte Carlo Simulation and for each of the ANN architectures described in Section 5.4.3.2. All KDE estimates were made using the value of $u(\omega)$ at the final time, $t = 1.0$.

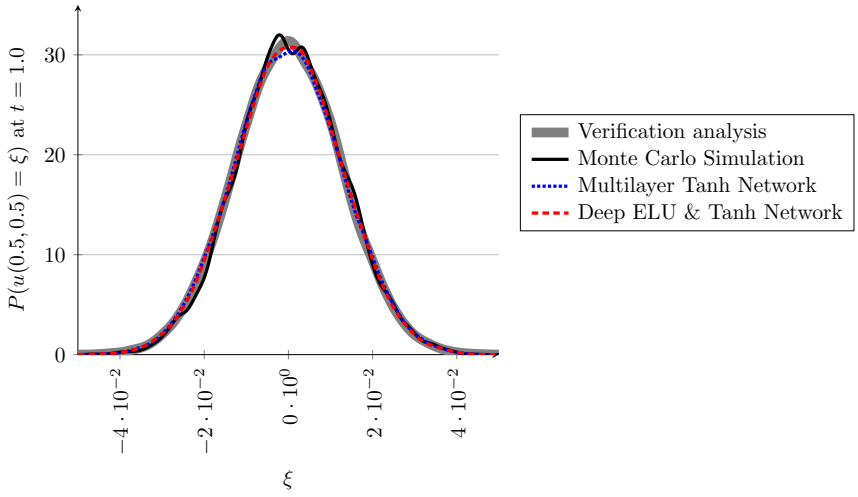
As in the first numerical experiment, the performance of each of the methods tested for estimating the probability density of $u(0.5, 0.5)$ (the centre of the spatial domain) at $t = 1.0$ is presented. The shape of the KDE estimators and their errors using 5000 known values (either training set size or Monte Carlo Simulations) is summarised in Figure 5.8. Figure 5.8a plots the Kernel Density Estimates for the verification analysis, $D_v(\omega = 0.5, 0.5, \xi)$, and for each of the methods tested. Figure 5.8b plots the error (as the difference) between the KDEs in 5.8a and the verification density. Table 5.2 presents the error $E(D_i, \omega = 0.5, 0.5)$ for each KDE D_i compared to the verification density, $D_v(\omega = 0.5, 0.5, \xi)$ at $t = 1.0$, for direct Monte Carlo Simulation and for each of the ANN architectures described in Section 5.4.3.2. Figure 5.9 plots the estimator error versus number of input values data from Table 5.2 for each of the methods tested.

Method	Type	KDE Error per number of training examples				
		Number of training examples				
		1000	2000	3000	4000	5000
MCS	N.A.	1.24082	0.75807	0.36820	0.21342	0.21013
ANN	ELU & Tanh	0.06893	0.03584	0.01574	0.04145	0.03398
ANN	ML Tanh	0.52755	0.38912	0.08152	0.08368	0.06613

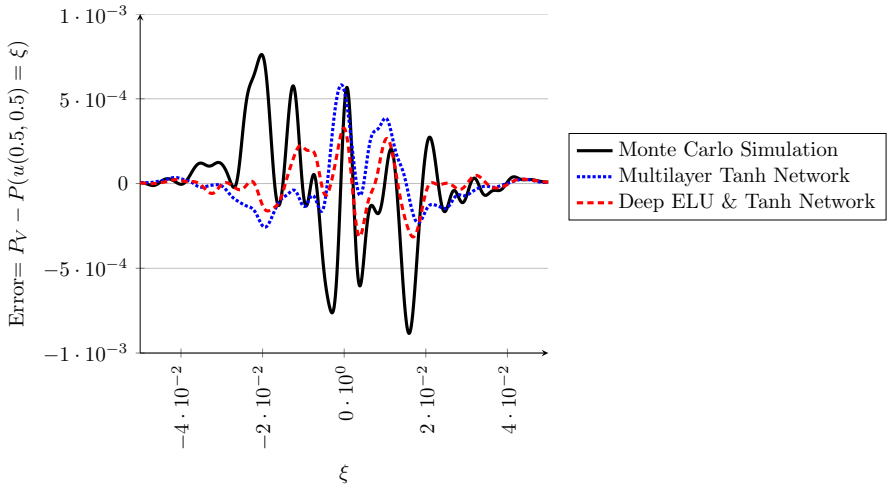
Table 5.2: Nonlinear heat equation problem Kernel Density Estimator (KDE) error for $u(0.5, 0.5)$ at $t = 1.0$. Errors are calculated for the specified number of training examples as the sum of squared errors relative to the verification KDE (from $1{\times}10^5$ MCS analyses) as per the description in Section 5.4.1.3. The method column refers to either direct Monte Carlo Simulation (MCS) or Artificial Neural Network (ANN) regression. Type N.A. denotes 'not applicable'. The type column for ANN methods refer to the ANN architectures in Section 5.4.3.2.

5.4.3.4 Discussion

This numerical experiment demonstrated that ANNs can successfully be used as fixed time surrogate models for nonlinear PDE initial value problems. From



(a) Kernel Density Estimates of $u(0.5, 0.5)$ at $t = 1.0$



(b) Errors of Kernel Density Estimates of $u(0.5, 0.5)$ at $t = 1.0$ compared to verification analysis.

Figure 5.8: Nonlinear heat equation problem Kernel Density Estimates and error for $u(0.5, 0.5)$ at $t = 1.0$. Figure 5.8a shows, for each method tested, the KDEs of $u(0.5, 0.5)$ which is the solution field of the probabilistic heat equation problem described in Section 5.4.3.1 at the centre of the spatial domain shown in Figure 5.5. Figure 5.8b shows the error of each of these plotted density estimates compared to the verification analysis KDE (from 1×10^5 direct Monte Carlo Simulations).

Figure 5.9 it is clear that while both types of ANNs tested were more accurate than direct MCS, the ELU rectified network consistently outperformed the tanh only network. Figure 5.8a demonstrates that both networks tested fit the verification analysis KDE distribution well but the ELU rectified network was more accurate, as shown in Figure 5.8b. The oscillations in the accuracy in the ELU network shown in Figure 5.9 are due to the fact that the network

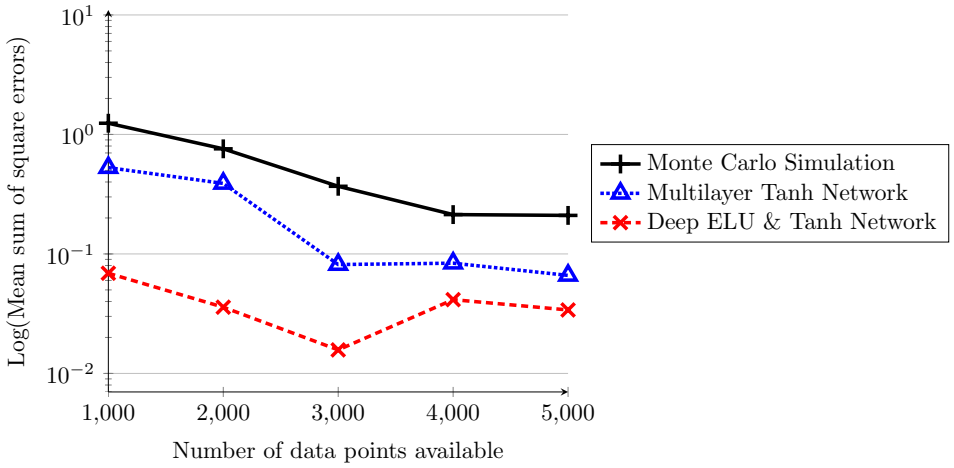


Figure 5.9: Comparison of errors for MCS and ANN-MCS using different ANN surrogate model architectures for the nonlinear heat equation problem at $t = 1.0$. Monte Carlo Simulation analyses were calculated directly using the the number of analyses shown on the horizontal axis. The ANN models tested were fit using training set of size shown on the horizontal axis. The ANN-MCS Kernel Density Estimates were built from $1{\times}10^5$ simulations from the surrogate models. Errors are calculated as the mean sum of squared differences between the Kernel Density Estimate using the different function approximation techniques and the verification analysis KDE (from $1{\times}10^5$ direct MCS analyses). Note that the errors were calculated with a fixed width Gaussian kernel for all analyses, as such the error shown should be considered relative to one another and not as absolute error indicators (which are kernel bandwidth dependent).

weights were reinitialised for all ANNs for each analysis.

Although it is not shown in the numerical results, it took some time to find a workable training schedule to adopt. Failure to use an adequate training schedule led to poor ANN convergence. Estimating a good training schedule for ANN-MCS will likely remain a difficult task for future research. It was found that waiting for the estimated training loss at a particular learning rate to begin to oscillate before reducing the learning rate by a factor of ten worked well in practice, as shown in the numerical results presented.

5.4.4 Sequence prediction for a nonlinear heat equation using RNN-MCS

The third numerical experiment presented introduces Recurrent Neural Networks for initial value PDE problem time sequence prediction. The nonlinear heat equation presented in Section 5.4.3 is reanalysed using the RNN-MCS method described in Section 5.3.3. This example demonstrates that it is possible to use RNNs for predicting the sequence of values output at discrete times by a initial value problem solver. This is in contrast to the second experiment which used an ANN to predict the solution field value at the final time only. Note that the transient nature of the equation solved is only dependent on the initial conditions. A more complex problem would be to deal with time dependent forcing. This would be a useful avenue for future research.

5.4.4.1 Problem definition

The equations analysed for this numerical problem are identical to those used for the second numerical experiment, described in Section 5.4.3.1. As in the second numerical experiment, time was taken to run from $t = 0$ to $t = 1$ in and discretised into 50 intervals.

For this numerical experiment, the goal is to estimate the probability density of $u(0.5, 0.5)$ (the central point) at every discrete time analysed given the probabilistic input random fields for the material coefficient, source and initial value fields denoted $g(\omega)$, $f(\omega)$ and $u_0(\omega)$ respectively. The probability distributions for the input fields are given in Section 5.4.1. With 51 time steps (50 steps plus $t = 0$) the discrete approximation $u(\omega)$ for all ω and all t is a sequence of 51 vectors. In particular, the aim is to train a RNN to predict sequences of $u(0.5, 0.5)$ for all t for a given input. As the experiment in this Chapter is a demonstration only, RNN-MCS was used to only predict the output values at the single location required rather than the full $u(\omega)$ field at all times.

5.4.4.2 RNN architecture and training details

A single Recurrent Neural Network architecture (see Section 5.4.1.5 for definitions) was used for this test case:

- *LSTM Network*: $\{I, D(I, L), M(I{\times}5), D(I{\times}5, E), M(I{\times}3), D(I, E), D(1, L)\}$

where where $I = 81{\times}3 = 243$ is the common input dimension. The input dimension is the sum of the dimensions of the coefficient vector field, $g(\omega)$, the source vector field, $f(\omega)$ and the solution field, $u(\omega)$. The output is one dimensional per time step ($O = 1$) for a total output dimension of $1{\times}51$. The network layers have widths of I, $I{\times}3 = 729$ and $I{\times}5 = 1215$. The network consists of alternating LSTM and ELU layers. The layers decrease in width from the input to the output. The LSTM and ELU combinations make this architecture is similar to the successful tanh & ELU architecture used for the second numerical experiment presented (see Section 5.4.3.2).

A training set sampler scaling factor of $\lambda = 1.1$ (see Section 5.4.1.4) was used for the RNN-MCS analysis. The RNN was trained using a reducing learning schedule for 50 epochs as follows:

1. $\eta = 1{\times}10^{-3}$ for 10 epochs
2. $\eta = 1{\times}10^{-4}$ for 10 epochs
3. $\eta = 1{\times}10^{-5}$ for 10 epochs
4. $\eta = 1{\times}10^{-6}$ for 10 epochs
5. $\eta = 1{\times}10^{-7}$ for 10 epochs

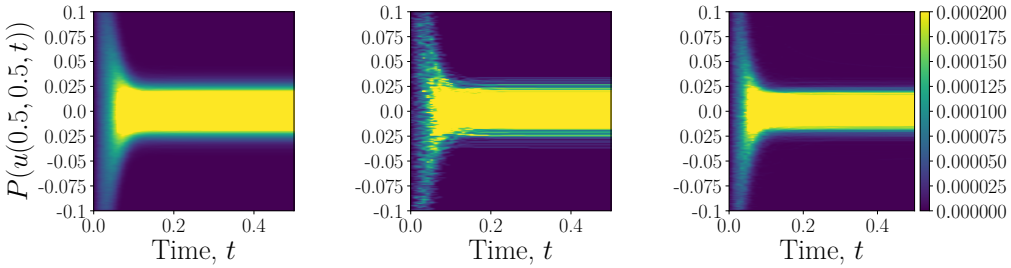
As with the other numerical experiments presented, this training schedule was found empirically and is based on a simplified form of simulated annealing (time decaying learning rate). Automating this process would be a useful avenue for future research.

5.4.4.3 Numerical results

Following the description in Section 5.4.1.2, $u(0.5, 0.5)$ trajectories from $t = 0.0$ to $t = 1.0$ at 51 evenly spaced time steps were calculated using both MCS and an RNN surrogate model. Trajectory probability densities using each method were calculated using a two dimensional histogram. The discrete time steps were used as histogram bins in the time dimension. The ξ dimension consisted of 202 bins per time with a width of 0.001 from -0.1 to 0.1 (200 bins) plus a $\xi \geq 0.1$ bin and a $\xi < -0.1$ bin. A verification analysis, MCS using $1{\times}10^5$ simulations, was carried out. The test MCS analysis histogram

was calculated using 1000 trajectory simulations. The RNN model was fit using 1000 training simulations and the RNN-MCS histogram was calculated from 1×10^4 integration trajectory predictions.

Histograms for each of the analyses are presented in Figure 5.10. Figures 5.10a, 5.10b and 5.10c show histograms for the verification, MCS and RNN-MCS tests respectively. The histogram error for the MCS and RNN-MCS tests was calculated by subtracting the calculated histograms from that for the verification analyses. Plots of these errors are shown in Figure 5.11 where Figure 5.11a and 5.11b are the errors histogram errors for the MCS and RNN-MCS tests respectively. Finally, the sum of squared errors for the MCS and RNN-MCS analysis at each time is shown in Figure 5.12. The sum of square error values shown were calculated using the histogram errors in Figure 5.11 at each discrete time step.

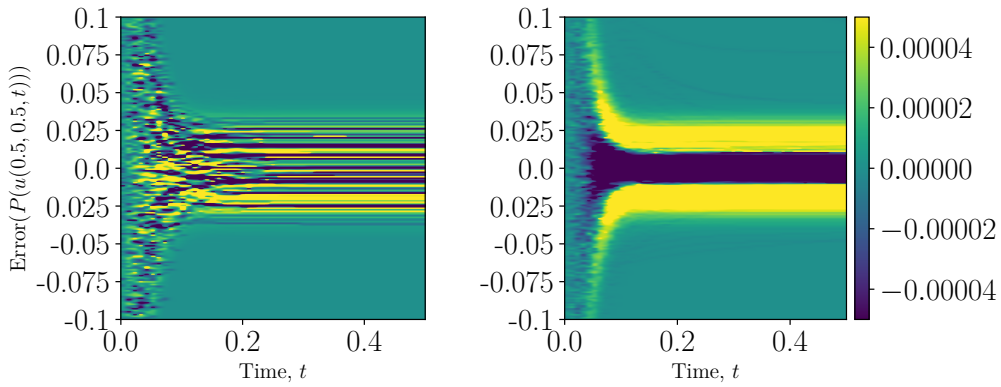


(a) Verification analysis (b) MCS histogram (c) RNN-MCS histogram

Figure 5.10: Nonlinear heat equation time sequence prediction problem histograms for $u(0.5, 0.5)$. $u(0.5, 0.5)$ is the solution to equation (5.41) at all time steps at the centre of the spatial domain shown in Figure 5.5. Histogram bins are calculated using discrete time steps from $t = 0$ to $t = 1.0$ in 50 steps (51 bins total) and for $u(0.5, 0.5)$ using 200 bins of width 0.001 from -0.1 to 0.1. Figure 5.10a shows the histogram for the verification analysis, 1×10^5 trajectories from a Monte Carlo Simulation. Figure 5.10b shows the histogram calculated from 1000 direct MCS trajectories. Figure 5.10c shows the histogram calculated from 1×10^4 predicted trajectories from the RNN LSTM surrogate model trained on 1000 known trajectories. The color bar is shared between all plots.

5.4.4.4 Discussion

This numerical experiment demonstrated that LSTM RNN networks can be used as time series prediction surrogate models. This was demonstrated for a nonlinear heat equation problem. It would be worthwhile, as a future application test, to extend the RNN surrogate model method to a more complicated



(a) MCS histogram error

(b) RNN-MCS histogram error

Figure 5.11: Nonlinear heat equation time sequence prediction problem histogram error for $u(0.5, 0.5)$. Figures 5.11a and 5.11b show the histogram error for MCS and RNN-MCS respectively. The histogram errors are calculated by subtracting the estimated histogram from the verification analysis histogram. Specifically, Figure 5.11a shows the histogram in Figure 5.10a minus that in Figure 5.10b. Figure 5.11b shows the histogram in Figure 5.10a minus that in Figure 5.10c. The color bar is shared between both plots.

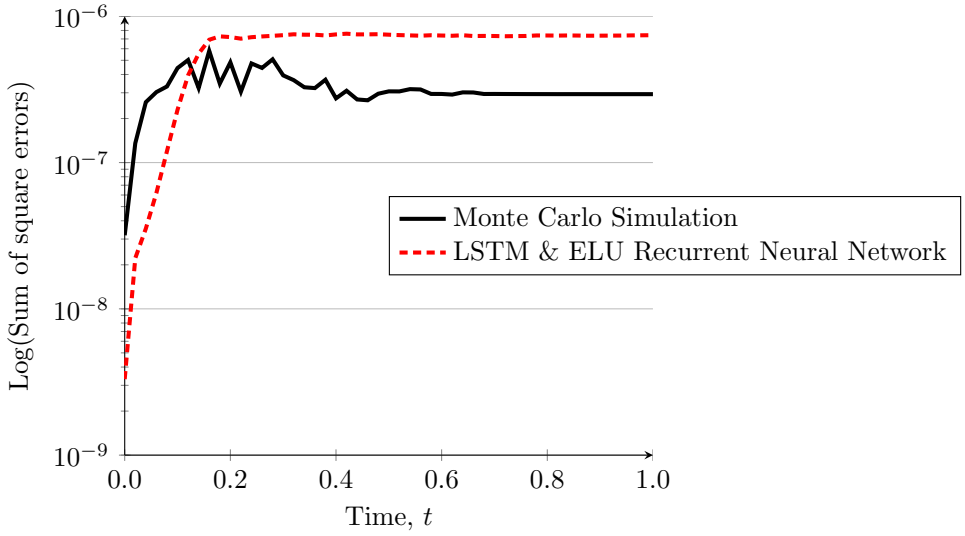


Figure 5.12: Nonlinear heat equation time sequence prediction problem histogram sum of squared errors versus time for direct MCS and LSTM RNN-MCS surrogate model. Errors are calculated using the sum of squares at each time step from the error plots shown in Figures 5.11a and 5.11b for MCS and RNN-MCS errors respectively.

hyperbolic PDE Uncertainty Quantification problem as well as problems with time dependent forcing. The histograms in Figure 5.10 demonstrate that the output distribution predicted by the RNN is very similar to the true distribu-

tion predicted by the verification analysis.

Interestingly, the RNN predicted output distribution was slightly too narrow compared to direct MCS and the verification analysis. The RNN-MCS test underestimates the variance of $u(0.5, 0.5)$ for times from around $t \approx 0.15$ onwards. The direct MCS error is fairly randomly distributed, as shown in Figure 5.11a. The RNN error, shown in Figure 5.11b, shows a different structure. In particular, the error RNN is less than the MCS error for early time steps. However, the RNN predicts too narrow a distribution after $t \approx 0.15$. This is the cause of the negative error values concentrated near $u(0.5, 0.5) = 0.0$ flanked by positive error values in Figure 5.11b. These observations are further supported by Figure 5.12 which shows that the RNN predicted output distribution demonstrated better convergence (lower error) than the MCS distribution for times up to around $t \approx 0.15$. After this point, MCS predicts the spread of values is predicted slightly more accurately than the RNN distribution. However, the errors of the MCS and RNN-MCS methods for later times are within the same order of magnitude.

Overall, the performance of the RNN surrogate model was quite good, predicting errors that were at best two orders or magnitude better than the direct MCS estimate. The worst case errors were not much different from the MCS errors. These errors were partly caused by the short training schedule (50 epochs). A longer training schedule would likely reduce the error of the RNN surrogate to below that of the MCS estimator for all time steps. A more intensive training schedule would be appropriate for a surrogate model that was to be used repeatedly in the long term, such as for weather prediction. Despite the slight narrowing of the predicted output distribution, the RNN surrogate model technique demonstrated could also be used to as a tool for estimating when, for example, a monitored physical system was likely to exceed acceptable tolerances by quickly predicting future performance based on current monitoring data.

It is important to note that the transient nature of the heat equation as modelled is purely due to the variability in the initial conditions. Time dependent stochastic source fields were not modelled, despite the fact that such models would be of practical utility. As these analyses are a first demonstration of the analysis concept, this more challenging problem has not yet been attempted. Further research will be required to evaluate the performance of the proposed

RNN-MCS method for problems with time dependent forcing.

5.5 Conclusions

This Chapter demonstrated that Artificial Neural Networks can be used to augment Uncertainty Quantification problems for Partial Differential equations by acting as surrogate models. It was shown that feedforward ANNs can be used as surrogate models for boundary value PDE problems and for static time prediction with initial value problems. In addition, it was shown that Recurrent Neural Networks can be used as time dependent surrogate models for initial value PDE problems by learning to generate a time-ordered sequence of values for a given PDE input. Uncertainty Quantification using these ANN and RNN surrogate models was demonstrated using finite element solutions for a linear boundary value problem and a nonlinear initial value problem.

The use of numerical simulation to model physical phenomena, in general, will contain errors associated to the choice of model. The choice of physical laws to be modelled, material constitutive equations, boundary conditions as well as the choice of numerical simulation procedure (for example, the Finite Element Method) are all potential sources of model error. Further inaccuracies may be introduced by approximation procedures used to find minimum residual solutions to some numerical approximation. The typical approach is to assume that these effective models are sufficiently accurate for some range of reasonably observable behaviour such that the simulation models can still be employed for future state prediction. Surrogate models of some physical phenomena based on observations of the solution of a numerical method will be at most as accurate as the numerical method used to approximate the actual physical situation. As such, the procedures described in this Chapter for accelerated Uncertainty Quantification are an additional source of model error. However, as the simulation model is available (unlike the hidden true physics of the world) it is possible to estimate the surrogate model error in terms of the loss function. The description of these errors clearly using language from supervised learning helps to render clear the accuracy of the surrogate model for Uncertainty Quantification problems.

The run time analysis presented in Section 5.3.4 discussed how the computa-

tional time complexity of surrogate model training should influence decisions on when to apply such models. In the numerical experiments analysed, the ANNs took some time to train but, once fitted, were able to produce predictions faster than the PDE solver for a given input. If a PDE function is to be reused repeatedly or if multiple different input distributions to a PDE are to be analysed, ANN based surrogate models are likely to be a worthwhile time investment. Note that for the analyses presented, the ANN training schedule was restricted to a fixed number of epochs. This was because of the comparative nature of the numerical studies in Section 5.4. If a trained model was to be used repeatedly, for example in the case of weather forecasting, it would be prudent not to artificially restrict the number of training epochs to be used for the network. The size of the networks could also be increased in practice for larger analyses than than those presented in this Chapter.

The ANNs demonstrated in this Chapter improved on earlier work in a number of ways. It was shown that by selecting an appropriate training schedule, it is possible to train many different network architectures for use as surrogate models. Deep networks were successfully trained as surrogate models by incorporating rectifier units (ReLU and ELU activations) and appropriate Stochastic Gradient Descent optimisation algorithms. It was demonstrated that the rectified deep ANN architectures outperformed traditional perceptron (sigmoid) style networks regardless of the widths or depths tested. Finally, it was shown that nonlinear time dependent functions can be learnt by using deep-in-time Recurrent Neural Networks with Long Short-Term Memory cells.

In the future, hopefully ANNs can be incorporated more widely into the Uncertainty Quantification tool kit along with existing, well-developed PDE surrogate model techniques such as Polynomial Chaos Expansions and Support Vector Regression. The surrogate model integration technique used in this Chapter, Monte Carlo integration, could be replaced by other methods. This would be an interesting subject for future research. Further, more test case analyses such as those presented here will help to uncover effective ANN architectures and training schedules appropriate for the various PDE problems encountered by numerical analysts. As automating ANN architecture design is a current area of research, the proof-of-concept analyses presented in this Chapter suggest that further progress could be made given effective adaptive

ANN architecture tools.

Chapter 6

A Bayesian interpretation of traditional Uncertainty Quantification techniques as Importance Sampling

Contents

Figures

Tables

Chapter 6 Overview

Key developments in Chapter 6 include:

- Section 6.2 explores a Bayesian interpretation of the solution of numerical equations through the lens of probabilistic numerical methods and analogies from Statistical Mechanics. The interpretation of the solution

of equations for Uncertainty Quantification as a type of joint input-output space marginalisation is discussed.

- Section 6.3 discusses background material on Importance Sampling.
- Section 6.4 provides an original contribution: that traditional approaches to Uncertainty Quantification based on pushforward measures are in fact a form of unnormalised Importance Sampling for which the normalisation constant is unjustifiably discarded.
- Section 6.5 presents numerical experiments to illuminate the theoretical material in the other parts of the Chapter.

6.1 Introduction

This Chapter takes recent developments in the Bayesian interpretation of equation solving and asks how the traditional approach based on residual minimisation can be understood from the Bayesian perspective. From the Bayesian viewpoint, as discussed in this Chapter, deterministic problems are in fact a special case of the broader set of probabilistic problems. As discussed in [175], typical numerical tasks such as optimisation and integration can be interpreted as Bayesian Inference problems. Moving from deterministic problems to Uncertainty Quantification, the main contribution of this Chapter is to demonstrate that Monte Carlo estimates of uncertain quantities by repeated function evaluation using residual minimisation is in fact an implicit form of Importance Sampling. By understanding equations from a Bayesian perspective, the variance of the implicit Importance Sampling estimate can be evaluated and as such the proposal sampling distribution modified depending on need. The variance of Importance Sampling estimates is dependent on the selected proposal distribution. The full Bayesian understanding of this Importance Sampling estimate makes clear the relevance of multimodality in the solution error space by identifying a likelihood probability term that is, in traditional techniques, simply assumed to be unity. By explicitly addressing this likelihood term, the weaknesses of traditional techniques can be mitigated and the ability to numerically evaluate uncertain output functions of an equation can be improved. It is noted here that in this Chapter residual minimisation refers to the solution of equations, not to finding fitted probability

density functions, which may still be evaluated by residual minimisation over the space of appropriate probability density functions.

To 'solve an equation numerically' is often taken to mean finding a solution that, within a certain tolerance, satisfies the constraints placed on the possible solutions by the form of the given equation. The constraints defining a solution are often specified directly in terms of an objective function. The term residual function is also used. A *numerical equation solver* can be taken to mean a program (which may be abstractly represented by just an algorithm) that takes a set of input values and produces (if possible) the minimum residual output. The criteria defining a 'good' solver may include the ability to find a global residual minimum and the speed at which the residual is reduced [178, 305, 148]. For physical problems, it is not easy to compare the performance of two different solvers on the same problem in the sense of whether or not the model is actually achieving a solution with the correct physical behaviour when the residual is minimised. For the purposes of this Chapter, it is assumed that the solvers used do not suffer from significant model error. This issue is discussed in more detail in Section 6.2. However, given a particular solver it is possible to ask if a solution is of a high quality or not based on the given residual.

Moving away from the single input, single output paradigm, it is becoming increasingly popular in science and engineering to ask probabilistic questions regarding the outputs of some equation. This can be framed in terms of Uncertainty Quantification [95, 21, 118, 351, 359, 356, 357, 40, 253], that is, given a set of uncertain inputs to an equation, what is the probability distribution over the space of outputs? The space of potential problems is incredibly vast. As single particular example from Civil Engineering, imagine a designer wishes to know the probability for the displacements of a structure to exceed some threshold value, for example in [141, 142, 94, 157]. The input strengths or potential loads applied to a structure may be unknown, but able to be estimated probabilistically. In this case, the physical model defines an energy function which constrains the relationship between the input and output space. As a more scientific example, an experimenter may wish to measure the power output from a laser and the variability in such a measurement [352].

Note that the examples typically raised in an Uncertainty Quantification context focus mainly on uncertainty in the input distribution. The probability

measure over the space of inputs to a model is assumed to include noise in the input data. Sources of input data noise could be low accurate measurement equipment or observations of complicated processes, such as chaotic dynamical systems. When carrying out numerical Uncertainty Quantification, it is sufficient to assume all sources of uncertainty are effectively factored into both the uncertain input distribution and the uncertainties in evaluation of the map from inputs to outputs. From the Uncertainty Quantification viewpoint, a 'good' solution can be considered to be an output distribution that is as close as possible (in a sense which can be quantified [300, 83, 225]) to the true distribution that is to be estimated. That is, the optimal encoding of the distribution over the solution space is desired. To this end, probabilistic problems are often interpreted as inverse problems. Considerable work has been done in advancing the interpretation of inverse problems from a Bayesian perspective [355, 257]. This interpretation of inverse problems considers that the error residual of a solution to a problem with a given input can be used to define a Gibbs measure where this error is taken to be a Hamiltonian (or energy functional).

The above two approaches to equation solving can be unified into a coherent framework. Understanding this unification provides the key to understanding the meaning of Uncertainty Quantification. Specifically, the probabilistic viewpoint subsumes the single solution case. The key principle is that the only indicator of the quality of a solution available when solving an equation is the residual function. For a given space of possible solutions (for example, $\mathbb{R}^n$) the residual function contains all of the constraints on the possible solutions and so fully specifies the structure of the equation to be solved. In this sense, which is detailed in Section 6.2, the residual function can be considered as a Hamiltonian on the solution space where high energy levels correspond directly to high residual values. Assuming that this residual function is fixed for a given problem, then the residual function can be considered to represent the Hamiltonian term in a canonical ensemble at a fixed inverse temperature, typically denoted β.

Given the representation of the residual function as a Hamiltonian, the Gibbs Measure can then be used to represent a probability distribution over the joint input and output space to the problem to be solved [355]. This is analogous to the log-linear model in machine learning [233, 323] and the Boltzmann Distri-

bution in statistical mechanics [284, 228]. In a thermodynamic canonical ensemble (by definition in thermodynamic equilibrium with a fixed temperature heat bath), the inverse temperature β is a Lagrange multiplier representing the accessibility of the various energy levels (§12 in [83]). In the probabilistic numerics framework presented here, β can be taken to represent the desired solution tolerance level. High β corresponds to low energy level accessibility, reducing the probability allocated to higher energy (high residual) states. The actual probability of the output distribution then can be calculated by assuming a prior distribution over the space of inputs and then conditioning the joint distribution of inputs and outputs by the input distribution. This is equivalent to marginalisation to extract the output probability distribution if the conditioning is taken over all possible inputs.

The meaning of a typical numerical procedure that produces a single output at a sufficiently low residual for a single input can then be understood in this way. The single input can be represented as a Dirac delta prior on the input distribution centred at that single point. Assuming that there is no prior over the output distribution, then the 'single output' is a Maximum Likelihood Estimate (MLE) of the output distribution. That is, the single solution is an estimate of the mode of the conditional output distribution [59]. This Bayesian viewpoint is in contrast with Fisher's original views on Maximum Likelihood which have been frequently criticised for a lack of consistency [8, 166]. The relationship between Maximum Likelihood and solving equations is discussed in detail in Section 6.2.

Typically, Uncertainty Quantification is performed by calculating the propagation of probability mass through the solution function to the output space. Specifically, a pushforward measure is implicitly calculated. Given these estimates of probability mass on the output space, a probability density function over the output distribution can be estimated based on the available data. There are two main components to this uncertainty propagation procedure, generating data on the output space and probability density estimation given this data. This is described in detail in Section 6.2. The inverse required to compute the pushforward measure is, in all but trivial cases, likely to be a multi-valued function. For physical problems, this causes a number of difficulties. Parameter identification by the solution of probabilistic inverse problems, for instance, is difficult if a solution is assumed to be single valued when it is

in fact multi-valued (that is, that there is no unique inverse). This suggests that a Bayesian approach based on measures of the proposed solution to some inverse function can be used to avoid issues of non-uniqueness. In particular, the solution of probabilistic problems can be interpreted as marginalisation over the joint input-output space of some physical model.

Direct Monte Carlo Simulation (MCS) is the simplest of the typical estimation procedures used [158, 118]. For example, the output probability density can be estimated by MCS by repeatedly sampling from the input distribution, then finding the solution of the problem using some given solver (for example a Finite Element Method representation of a partial differential equation). These sampled solutions can then be used to build a representation of the output distribution. Examples include simple histograms [224] or a more sophisticated series expansion in some optimal basis, as in Polynomial Chaos methods [141, 142, 254]. Direct Monte Carlo can be augmented by Markov Chain Monte Carlo [313, 55, 157], Karhunen-Loève expansion (as in the Spectral Stochastic Finite Element Method) [141] or any of the assortment of numerical techniques that have been developed for Uncertainty Quantification [351]. Traditionally, these uncertainty propagation techniques have been used to calculate the output distribution directly, minimising the residual error in what is in effect a series of intermediate MLE evaluations that are used to estimate the output distribution.

From the Bayesian perspective, it is clear that Uncertainty Quantification techniques that rely on residual minimisation follow a three step process. First, a residual function is constructed that is a functional of the input given the output. Second, an optimisation procedure is applied to find an MLE that is within a specified tolerance. Finally, a Dirac delta distribution is implicitly assumed that trivialises the marginalisation over the joint input-output distribution by implicitly setting the integral equal to the residual evaluated at the single non-zero input point. Probabilistic sampling estimates calculated in this way are in fact an implicit form of Importance Sampling that use a proposal distribution based on Maximum Likelihood Estimates. Demonstrating that this is indeed the case and exploring the consequences of this choice of proposal distribution is the primary contribution of this Chapter. Common objective optimisation procedures include assorted forms of function inversion, such as solving matrix equations [148], simulated annealing [305], genetic algo-

rithms [102] and stochastic gradient descent [347] and several others [178, 305]. The MLE produced would be a Maximum a Posteriori (MAP) estimate if a prior distribution over the output space was used. In the remainder of the Chapter, it is assumed that the description of the final phase of traditional optimisation as an MLE, rather than a MAP, is without loss of generality. This implicit MLE based approach to solving equations is not useful from an Uncertainty Quantification perspective. An MLE may not be desirable, especially if there are degenerate ground state solutions present and/or the residual is unimodal. Instead, one may wish to calculate a probability that is a function over the space of possible outputs. As a simple but illustrative example, the probability that a spring with a variable stiffness under random loading exceeds a certain strain cannot be calculated simply by an MLE. Further, if a distribution more complicated than a Dirac delta is used to model the input space, as is the case in Uncertainty Quantification problems, it is no longer possible to ignore the effect of the input distribution in reweighting the probability mass of the conditional output distribution.

Given the Bayesian interpretation of problem solving it is interesting that the traditional methods of Uncertainty Quantification solving often work quite well. In particular, traditional methods can be highly effective for estimating the mean of an output distribution [118] but fail for the rare event estimation that is often required for reliability analysis [15, 16, 290, 157, 13]. This Chapter demonstrates that the traditional methods for Uncertainty Quantification can be interpreted as Importance Sampling with the possibly unwarranted assumption that the conditional probability of the solution given the input is unity. Importance Sampling refers to the density estimation of a probability distribution by considering the ratio of the distribution to be estimated with a different distribution called the proposal distribution [314, 320]. Sampling from this proposal distribution is used calculate random variable expectations.

The utility of Importance Sampling is that often the proposal distribution can be selected such that it is of a simpler structure in some sense than the original distribution and so that the variance in an estimated functional of the original distribution is reduced. The variance reduction of Importance Sampling is heavily dependent on the selected proposal distribution. A poor selection can actually increase the variance in an estimated quantity. By understanding the

implicit proposal distribution used in traditional Uncertainty Quantification, new proposal distributions can be constructed that better achieve better variance reduction for the quantity that is to be estimated. Further, the success of traditional Uncertainty Quantification for certain problems, such as those with unimodal input distributions coupled with unimodal residual functions at high tolerance, can be understood by realising that the implicit proposal distribution happens to have a low variance for the estimate of the mean of the output distribution. This is detailed in Section 6.4.

Finally, in Section 6.5, several numerical experiments are presented to support the theoretical developments of this Chapter. The first experiment demonstrates and explains the failure of the traditional forward approach for far from mean threshold estimation problems. A multivariate Gaussian, for which very high accuracy probability estimates can be found independently, is used to describe the joint input-output probability space for an Uncertainty Quantification problem. By calculating the true variance of a Monte Carlo Simulation estimate using the full Bayesian approach, it is clear that the implicit MLE Importance Sampler (i.e. the traditional forward approach) has high variance which does not reduce with further simulation iterations. This suggests that an improved sampling regime can be designed based on an understanding of the variance Importance Sampling due to a selected proposal distribution (discussed in Section 6.3) and the Bayesian interpretation of the residual function as a conditional probability on the joint input-output space. The second numerical experiment supports the first by introducing a bimodal residual surface for a forward problem where the quantity to be estimated is far from the global minimum. It is shown that by including the correct Bayesian variance estimate, an improved proposal distribution can be designed to estimate quantities for which the traditional residual minimisation methods fail. Finally, a third numerical experiment is presented that explores the effect of multimodality of the residual function and demonstrates that Uncertainty Quantification can be carried out without finding the minima of the problem residual function. Specifically, it is shown that Markov Chain Monte Carlo sampling can be used directly on the joint input-output probability space. These numerical results open the way for future work in the definition of improved sampling regimes for Monte Carlo based Uncertainty Quantification.

In general, any simulation model based on mapping observations to an alter-

native function space can be reasoned about in the abstract by the approach discussed in this Chapter. This encompasses Civil Engineering numerical Uncertainty Quantification problems, which map observations (such as strains, mass flow rates and temperatures) to output spaces (such as stresses and material property parameters) via simulation techniques including the Finite Element and Finite Volume Methods. In particular, as Uncertainty Quantification is becoming an increasingly popular approach to deal with problems in Civil Engineering, the abstract techniques discussed in this Chapter may lead to applications as a part of future research.

6.2 Interpretations of solving equations and Uncertainty Quantification

6.2.1 Preamble

For notational consistency, this Section defines several terms before the main result of this Chapter is presented in Section 6.4. The Bayesian and function inversion paradigms for Uncertainty Quantification are detailed. The Bayesian interpretation of Uncertainty Quantification starts by reframing the problem error residual functional as a probability distribution via the Gibbs measure [355, 228]. A full mathematical treatment of when this is possible is given in [355]. It suffices to say here that the conditions under which this identification is possible are sufficiently broad that the vast majority of numerical problems can be considered under this framework. The residual functions in [355] are taken over a Banach space [309], as this is captures both the $\mathcal{L}^1$ and $\mathcal{L}^2$ norms. As these are the most common forms of residual function used in practice, it is sufficient for the discussion here. Traditional Uncertainty Quantification techniques using residual minimisation are demonstrated to be a Maximum Likelihood Estimate on the solution probability space. Further, it is argued that the numerical solution tolerances and deterministic problems from function inversion theory can be subsumed by the Bayesian interpretation of solving equations. Finally, Uncertainty Quantification is defined. After having gone through these necessary preliminaries, the main result is given in Section 6.4. It is again noted here that in this Chapter residual minimisation refers to the solution of equations, not to finding fitted probability density functions,

which may still be evaluated by residual minimisation over the space of appropriate probability density functions. This will not be detailed further in this Chapter, but it is will be clear from the discussion in Section 6.2.3 that selecting a single fitted probability density estimate by residual minimisation over probability densities is also a Maximum Likelihood Estimate.

6.2.1.1 Input, output and residual functional distributions

The input and output spaces to a problem represent the domains on from which full problem specifications and solutions can be drawn. For example, the input space for an uncertain heat equation might be the selection of thermal conductivities at every point within the spatial problem domain. The output space would then, in this example, represent a possible solution defined at every point within the problem spatial domain.

Definition 6.2.1. *Input and output space*: Let the sets X and Y be the *input space* and *output space* respectively. Let points within the input and output spaces be represented by $x \in X$ and $y \in Y$ respectively. △

Definition 6.2.2. *Input, output and joint input-output probabilities*: Given X and Y, let $P(X)$ and $P(Y)$ be the *input probability density* and *output probabilities* respectively. Then let $P(X, Y)$ be the *joint input-output probability density*. △

Definition 6.2.3. *Residual functional*: Given a function space $\mathcal{H}$ let:

$$H : X \times Y \longrightarrow \mathbb{R}^+ \tag{6.1}$$

such that

$$H(y|x) \mapsto \epsilon \in \mathbb{R}^+ \tag{6.2}$$

be the *residual functional*. Other names for this functional include the *error*, *loss* and *energy* functional. The *Hamiltonian* is also used to refer to equation (6.1) in this Chapter. △

Definition 6.2.4. *Conditional output probability density*: Given X, Y and H, that $E(y|x)$ represents the *self-information* or *surprisal* [83] associated with a

value $y \in Y$ given $x \in X$:

$$H(y|x) = -\ln(P(y|x)) \tag{6.3}$$

Further, let the *conditional output* probability density be given by the Gibbs Measure (see [228]):

$$P(y|x) = \frac{e^{-\beta H(y|x)}}{Z_{XY}} \tag{6.4}$$

where Z_{XY} is the *normalisation constant* or *partition function*:

$$Z_{XY} = \sum_{x \in X} \sum_{x \in X} e^{-\beta H(y|x)} \tag{6.5}$$

and β is the *tolerance parameter*.

The distribution $P(y|x)$ will also be referred to as the *residual probability density* in this Chapter. △

6.2.2 The Bayesian interpretation of solving equations

The law of total probability (see Definition 2.5.2) can be used to calculate the probability density on the output space given a probability density $P(X)$ on the input space.

Definition 6.2.5. *Output probability density*: Given the *input probability density* $P(x)$ $\forall x \in X$, the *output probability density*, $P(y)$ $\forall y \in Y$, is calculated by the law of total probability (marginalisation):

$$P(y) = \sum_{x \in X} P(y|x)P(x) \tag{6.6}$$

△

The output probability density can be interpreted by considering Bayes theorem:

$$P(m|d) = \frac{P(d|m)P(m)}{P(d)} \tag{6.7}$$

where m represents a *model* of the *data*, d, and states that the *posterior probability*, $P(m|d)$, of a model given data depends on the conditional probability of the data given the model, $P(d|m)$, and the prior distribution, $P(m)$.

For Uncertainty Quantification, the Bayesian interpretation starts by considering the input probability distribution as defining a prior, $P(X)$. The conditional output probability density $P(Y|X)$ (fixed by the Hamiltonian) defines the conditional expectations required to calculate the probability of an output for a given input. This induces an inverse function residual (detailed in Section 6.2.4) which defines the solutions of inverse problems by $H(x|y) = \|x - x'(y)\|$ [355, 257]. As in the case of the $H(y|x)$, $H(x|y)$ also defines a probability density, $P(X|Y)$, by the Gibbs measure. These conditional probabilities can be combined, by considering Bayes theorem, into an expression for the joint input-output probability:

$$\begin{aligned} P(X|Y) &= \frac{P(Y|X)P(X)}{P(Y)} \\ P(X|Y)P(Y) &= P(Y|X)P(X) \\ P(X,Y) &= P(Y|X)P(X) \end{aligned}$$

where the final step above used the fact that the *joint input-output probability* $P(X,Y)$ is equal to $P(Y|X)P(X)$ [224]. Finally, the Bayesian interpretation of the output probability distribution is that $P(Y)$ is the marginal probability of y from $P(X,Y)$ given by equation (6.6).

From Definition 6.2.4 it is clear that the specification of the equation to be solved is essentially contained in the Hamiltonian and is independent of the choice of prior distribution $P(X)$. The selection of $P(X)$ can be made based on, for example, measurement data. Further, it is noted here that the selection of a model as $H(y|x)$ is actually a delta prior over the space of representable models. Expanding this definition would allow for information from multiple models to be combined. For more detailed philosophical and mathematical discussion on prior selection see [241, 346].

6.2.3 The relationship between energy and probability

The relationship between energy and probability is simply that increasing energy indicates decreasing probability and vice versa. From definition 6.2.4,

consider:

$$\begin{aligned} P(y|x) &= \frac{e^{-\beta H(y|x)}}{Z_{Y|X}} \\ -\ln P(y|x) &= \beta H(y|x) - \ln Z_{Y|X} \\ \beta H(y|x) &= -\ln P(y|x) + \ln Z_{Y|X} \\ \beta H(y|x) &= -\ln P(y|x) + F(y|x) \end{aligned}$$

where $F(y|x) := \ln Z_{Y|X}$ is termed the *variational free energy* [123]. The variational free energy is positive for all x, y.

From these equations, note that $H(y|x) \geq 0$ and $P(y|x) \in [0, 1]$ by definition. Then, the variational free energy is always positive:

$$F(y|x) = \beta H(y|x) - \ln P(y|x) \geq 0$$

Note that, for fixed $F(y|x)$:

$$\nabla \beta H(y|x) = \nabla \left(-\ln P(y|x) + F(y|x)\right) \tag{6.8}$$

$$\beta \nabla H(y|x) = -\nabla \ln P(y|x) + \nabla F(y|x) \tag{6.9}$$

$$\beta \nabla H(y|x) \propto -\frac{\nabla P(y|x)}{P(y|x)}$$

$$\beta \nabla H(y|x) \propto -\nabla P(y|x) \tag{6.10}$$

so $\beta \nabla H(y|x)$ decreases as $-\ln P(y|x)$ increases and as $P(y|x)$ decreases.

Further, $-\ln P(y|x) \geq 0$, and monotonically increases as $P(y|x)$ decreases (by the properties of ln). Finally, $\lim_{P(y|x)\to 0} [-\ln P(y|x)] = \infty$ as $-\ln [P(y|x) = 1] = 0$. Then $-\ln P(y|x)$ is an upper bound on $\beta H(y|x)$:

$$\begin{aligned} \beta H(y|x) &= -\ln P(y|x) + F(y|x) \\ \beta H(y|x) &\geq -\ln P(y|x) \\ \beta H(y|x) &\geq -\ln P(y|x) \geq -\ln [P(y|x) = 1] \end{aligned}$$

This shows that the minimum energy is lower bounded by the monotonic log probability and as such the minimum energy coincides with the maximum

probability for fixed β, $\forall x \in X$:

$$\underset{y \in Y}{\operatorname{argmin}} \; H(y|x) = \underset{y \in Y}{\operatorname{argmax}} \; P(y|x) \tag{6.11}$$

6.2.3.1 Likelihood and parameterised distributions

The conditional probability of an event given another is defined to be the *likelihood* of the parameter [59, 323]:

Definition 6.2.6. *Likelihood*: Given $P(a, b)$ for $a \in A$ and $b \in B$:

$$\mathcal{L}(b; a) := P(a|b) \tag{6.12}$$

$\triangle$

Maximum likelihood estimation refers to selecting the conditioning event with the maximum conditional probability [59, 323]:

Definition 6.2.7. *Maximum Likelihood Estimate (MLE)*: Given $P(a, b)$ for $a \in A$ and $b \in B$:

$$b_{mle} := \underset{b \in B}{\operatorname{argmax}} \left[\mathcal{L}(b; a)\right] = \underset{b \in B}{\operatorname{argmax}} \left[P(a|b)\right] \tag{6.13}$$

$\triangle$

The argmax operator in equation (6.13) is defined in the following Section in Definition 6.2.8.

If a particular prior distribution is assumed over $P(b)$, the MLE becomes a *Maximum a Posteriori (MAP)* estimate. Only the MLE case is considered in this Chapter, but there is no significant loss of generality of the results presented as the MAP case can be found from the MLE case by Bayes rule. More detail on MAP and MLE's can be found in [275].

6.2.3.2 The argmin and argmax operators and their inverses

It will be useful, before discussing an interpretation of the standard problem solving paradigms to carefully define the argmax and argmin operators. These operators return the input to a function that returns the maximum or minimum respectively of an evaluation of that function and are defined as

follows.

Definition 6.2.8. argmax (and argmin) operator: Given the set of maps $f \in B$ with $f : A \to \mathbb{R}^+$. Then the *argmax* of f over $A' \subseteq A$ is defined by the map:

$$\text{argmax} : B \longrightarrow \mathcal{P}(A) \tag{6.14}$$

such that:

$$\underset{a \in A' \subseteq A}{\text{argmax}} f(a) := \left\{ a | a \in A' \wedge \forall a' \in A' : f(a') \leq f(a) \right\} \tag{6.15}$$

Similarly, the *argmin* of f over A' is defined by the map:

$$\text{argmin} : B \longrightarrow \mathcal{P}(A) \tag{6.16}$$

such that:

$$\underset{a \in A' \subseteq A}{\text{argmin}} f(a) := \left\{ a | a \in A' \wedge \forall a' \in A' : f(a') \geq f(a) \right\} \tag{6.17}$$

$\triangle$

It will also be useful to define the inverse of argmax and argmin. Simply, the inverse of the argmax and argmin operators of $f : A \to \mathbb{R}^+$ is the set of all points in A that would return the argmax or argmin for $b = f(a)$ for $b \in B$.

With reference to Definition 6.2.8, consider the set of all maps in some function space from A to $\mathbb{R}^+$:

$$B := [f : A \longrightarrow B]$$

Then, the argmax (or argmin) are maps from B to the power set (set of all subsets) of A as the output value of the argmax (argmin) operators is in some subset of A:

$$\begin{aligned} \text{argmax} : \ & B \longrightarrow \mathcal{P}(A) \\ \text{argmin} : \ & B \longrightarrow \mathcal{P}(A) \end{aligned}$$

that is

$$\underset{a \in A' \subseteq A}{\text{argmax}}(f(a) = b) \mapsto a \in A' \in \mathcal{P}(A)$$

Then the inverse of argmax (argmin) is then defined by the map to subsets $B' \subseteq B \in \mathcal{P}(B)$:

$$\text{argmax}^{-1} : \mathcal{P}(A) \longrightarrow \mathcal{P}(B)$$
$$\text{argmin}^{-1} : \mathcal{P}(A) \longrightarrow \mathcal{P}(B)$$

Definition 6.2.9. Inverse argmax (and argmin) operator: Given the set of maps $f \in B$ with $f : A \to \mathbb{R}^+$. Then the *inverse argmax operator* for f over $A' \subseteq A$ is given by the map:

$$\text{argmax}^{-1} : \mathcal{P}(A) \longrightarrow \mathcal{P}(B)$$

such that, for some $a \in \hat{A} \subseteq A$:

$$\underset{a \in A' \subseteq A}{\text{argmax}}^{-1}(a) = \left\{ b \in B : \underset{a \in A' \subseteq A}{\text{argmax}}(b(a)) = a \right\}$$

Similarly, the *inverse argmin operator* is given by the map:

$$\text{argmin}^{-1} : \mathcal{P}(A) \longrightarrow \mathcal{P}(B)$$

such that, for some $a \in \hat{A} \subseteq A$:

$$\underset{a \in A' \subseteq A}{\text{argmin}}^{-1}(a) = \left\{ b \in B : \underset{a \in A' \subseteq A}{\text{argmin}}(b(a)) = a \right\}$$

△

6.2.3.3 MLE and the energy-probability relationship

Having defined terms, the relationship between energy and probability in the context of Uncertainty Quantification can be further explored. From equation (6.11), the minimum energy point in the output space is equivalent to the maximum probability point in the output space. In combination with Defi-

nition 6.2.6 of likelihood and equation (6.13), which defines MLE's, it can be shown that the argmax operator is simply the MLE. Given a fixed $x = x'$, let $P(y|x = x', \theta)$ be a parametrised probability distribution (in the sense of [123]) with parameters $\theta \in \Theta$ so:

$$\theta_{mle} = \underset{\theta \in \Theta}{\operatorname{argmax}} \mathcal{L}(\theta; y, x = x')$$
$$\theta_{mle} = \underset{\theta \in \Theta}{\operatorname{argmax}} P(y|x = x', \theta)$$

Then, if the parameters θ are taken so that $\Theta = Y$, then $\theta = y'(x') \in Y$ so that:

$$\mathcal{L}(y'; x = x', y) = P(y|x = x', y'(x')) := P(y = y'(x')|x = x') \tag{6.18}$$

then

$$\theta_{mle} = \underset{\theta \in \Theta}{\operatorname{argmax}} P(y|x = x', \theta) \tag{6.19}$$
$$\theta_{mle} = \underset{y'(x') \in Y}{\operatorname{argmax}} P(y = y'(x')|x = x') \tag{6.20}$$

so then the MLE parametrised can be identified with $y(x)$ by:

$$\theta_{mle} = y'(x')_{mle} \tag{6.21}$$

Following equation (6.21), the argmax operator can be expressed as an MLE:

$$y'(x)_{mle} := \underset{y'(x) \in Y}{\operatorname{argmax}} P(y = y'(x)|x) = \underset{y \in Y}{\operatorname{argmax}} P(y|x) \tag{6.22}$$

Using equation (6.22), consider then that the minimum energy output can be considered to be a maximum likelihood estimate of the output distribution:

$$\underset{y \in Y}{\operatorname{argmin}} H(y|x) = \underset{y \in Y}{\operatorname{argmax}} P(y|x) = y'(x)_{mle} \tag{6.23}$$

Further, consider that often the inverse of a function is calculated by the minimisation of some residual, $H(x|y)$, for example in iterative matrix inversion [148]. With arguments identical to those for the forward residual minimisation

problem, letting $\Theta = X$:

$$x'(y)_{mle} := \underset{x \in X}{\operatorname{argmin}}\ H(x|y) = \underset{x \in X}{\operatorname{argmax}}\ P(x|y) \tag{6.24}$$

As MLE's find the minimal value of residual functions, they are the key part in understanding the traditional solution of forward and inverse problems as discussed in the following section.

6.2.4 Interpreting standard methods of solving equations

It will be shown that both the forward and inverse techniques can be understood in terms of the Bayesian perspective by MLE's and the pushforward measure. Standard or traditional Uncertainty Quantification methods use a either a forward or inverse problem paradigm to estimate $P(Y)$. The inverse problem framework calculates $P(Y)$ by pushing forward (in the sense of pushforward measure, [50, 219] and Definition 6.2.12) the input distribution to the output space. Then both the forward and inverse problems rely on a change of variables formula derived from the pushforward measure (Definition 6.2.13).

In both cases, the traditional approach to problem solving uses MLE's to evaluate both forward and inverse values of functions. It will be shown in Section 6.4 that this is a form of Importance Sampling with an additional unjustified assumption on the likelihood of parameters. By contrast, Bayesian methods make use of the conditional output probability density in combination with the input density to find $P(Y)$. Fundamentally, the Bayesian viewpoint is that function inversion for the solution of equations defined by a residual is subsumed by the understanding that the residual must be specified over both the input and output spaces simultaneously and as such there is a joint probability over both of these spaces that should be accounted for. By detailing the standard forward and inverse solution procedures, the need in Section 6.4 for the Bayesian approach will become clear as it enables handling of the likelihood term that is ignored by the standard approach. This section, after defining pushforward measures, gives a definition of forward and inverse problems based the understanding of MLEs given in Section 6.2.3.

6.2.4.1 The solution of forward and inverse problems

The solution of a problem is typically defined from a standard perspective on solving equations as residual minimisation. However, from the discussion in Section 6.2.3.3, residual minimisation can be interpreted as finding MLE's.

Definition 6.2.10. *Forward problem solution:* Given input and output spaces X and Y, a function $u : X \to Y$ and residual function $H : X \times Y \to \mathbb{R}^+$, the *forward problem solution* is defined as:

$$u(x) := \underset{y \in Y}{\operatorname{argmin}}\ H(y|x) \tag{6.25}$$

△

From equation (6.22), the forward problem solution has a direct interpretation as an MLE for a given x:

$$y'(x)_{mle} = \underset{y \in Y}{\operatorname{argmin}}\ H(y|x) = u(x) \tag{6.26}$$

An inverse problem is defined similarly, although more care must be taken. First, notice that from Definition 6.2.10 u can be considered a map:

$$u : X \xrightarrow{u_1} \mathcal{H} \xrightarrow{u_2} Y \tag{6.27}$$

This is because $u(x)$ first selects a residual $H(\circ|x)$ and the evaluates the argmin on $H(\circ|x)$ over Y. Then the definition of the inverse of u must be such that this process is reversed:

$$u^{-1} : \mathcal{P}(Y) \xrightarrow{u_2^{-1}} \mathcal{P}(\mathcal{H}) \xrightarrow{u_1^{-1}} \mathcal{P}(X) \tag{6.28}$$

With reference to Definition 6.2.9, the first part of u^{-1} is the inverse of the argmin which will map from $\mathcal{P}(Y)$ to $\mathcal{H}(\circ|X)$. Call this map u_2^{-1}. Denote the remaining part of the inverse by the map $u_1^{-1} : \mathcal{P}(\mathcal{H}) \to \mathcal{P}(X)$. Then $u_1^{-1}(H(\circ|x))$ must find all $x \in X$ such that $H(\circ|x) \in u_2^{-1}(y)$.

In other words, the inverse problem solution finds the input value set $X' \subseteq X$ such that, given an output, $y_m \in Y$, each $x' \in X'$ evaluates $H(y|x)$ to the minimum value, $\min H(y = y_m|x)\ \forall x' \in X'$. Intuitively, the inverse

solution finds all of the input values that, given a particular output, evaluate the residual function to its minimum value. Given y', any selected single value $x' \in X'$ should evaluate to the correct residual.

Definition 6.2.11. *Inverse problem solution:* Given input and output spaces X and Y, a residual function $H \in \mathcal{H}$ such that $H : X \times Y \to \mathbb{R}^+$, the *inverse problem solution* is defined as the map:

$$u^{-1}(y) : Y \to \mathcal{P}(\mathcal{H}) \to \mathcal{P}(X) \tag{6.29}$$

such that

$$u^{-1}(y) := X'(y) = \left[\underset{y \in Y}{\operatorname{argmin}} \; H(y|x) \right]^{-1} (y) \tag{6.30}$$

△

Inverse problems are often expressed in terms of residual minimisation. The iterative solution of matrix problems is a common example [178, 148]. Defining inverse problems in terms of residual minimisation induces a new residual, $H(x|y)$, which gives the error of the evaluated MLE for x' from the true residual given y' and x':

$$H(x = x'|y = y') = \|x' - u^{-1}(y')\| \tag{6.31}$$

Equation (6.31) reveals how inverse problems are also solved by Maximum Likelihood Estimates. By a virtually identical argument for that used to find equation (6.23), the solution set X' is populated by MLE's of $H(x|y)$ because $H(x|y)$ also defines a probability distribution by the Gibbs measure as in Definition 6.2.4:

$$\underset{x \in X}{\operatorname{argmin}} \; H(x|y) = \underset{y \in Y}{\operatorname{argmax}} \; P(x|y) = x'(y)_{mle} \tag{6.32}$$

Then, the solution of the inverse problem is given by the set $X'_{mle}(y)$ of all MLE's, $x'(y)_{mle}$, of the inverse function:

$$X'_{mle}(y) = \left\{ x' \in X' \subseteq X | x' = x'(y)_{mle} \right\} \tag{6.33}$$

If it is assumed that there is a single inverse, (as is often the case for numerical problems [373]) then the MLE inverse solution reduces to a selection $x'_{mle} \in$

X'_{mle} so that

$$u^{-1}(y) = x'(y)_{mle} \sim \left[\underset{y \in Y}{\operatorname{argmin}}\ H(y|x)\right]^{-1} \tag{6.34}$$

Solving problems by function inversion is typically ill-posed [372]. The reason for this ill-posedness is highlighted by equations (6.29) and (6.30). Inversion of the argmin operator is non-surjective as the preimage of this operator is a region $X' \in X$. Further, the second inversion from the function space $\mathcal{H}$ to X is also possibly non-surjective. Traditional inversion solutions, as discussed, effectively make MLE's to evaluate the required inverses. These MLE's are often framed in terms of regularisation, for example Tikhonov regularisation [372, 373], which approximates inverse problem solutions by a single value using the modified residual. For example, in a linear matrix inversion problem:

$$\|Ax - b\| \tag{6.35}$$

can be augmented by the Tikhonov matrix, Γ, such that

$$\|Ax - b\| + \|\Gamma x\| \tag{6.36}$$

yields unique solutions

$$\hat{x} = (A^T A + \Gamma^T \Gamma)^{-1} A^T b \tag{6.37}$$

The effect of this regularisation is to prefer certain solutions within the set of inverse solutions, allowing for easier numerical evaluation. Extensions of Tikhonov Regularisation to nonlinear cases are described in [370]. Further, there is a direct interpretation of regularisation in terms of Bayesian linear regression and prior distributions, as discussed in [275].

It is clear that for the forward and inverse problems (Definitions (6.2.10) and (6.2.11) respectively), that MLE's are used to find solutions that are optimal in the sense of a minimising residuals. It is demonstrated in Section 6.3.4 that the use of MLE's in this way is implicitly a form of Importance Sampling that assumes a certain likelihood term can be ignored. The Bayesian viewpoint is that these MLE's are not necessarily required to find a solution to a problem and in fact may introduce errors if the relative likelihood of these MLE's is not calculated, particularly for multimodal solutions. This is made more clear by

considering probabilistic estimation using the traditional forward and inverse problem viewpoints, which first requires an understanding of the pushforward measure.

6.2.4.2 Pushforward measures and change of variables

The pushforward measure is a technique for, in a sense, transferring a measure from one space to another. The pushforward measure allows for a particular change of variables for a probabilistic function to be expressed in terms of function inverses. These definitions will allow for the traditional methods of probabilistic analysis of equations to be understood in terms of MLE's. The pushforward measure is defined from Proposition 3.2.1 in [219] as:

Definition 6.2.12. *Pushforward measure:* Given measurable spaces (A, Σ_A) and (B, Σ_B), a measurable map $f : A \to B$ and a measure $\mu : \Sigma_A \to [0, +\infty]$, the *pushforward of* μ is the measure $f_*(\mu) : \Sigma_B \to [0, +\infty]$ given by:

$$(f_*(\mu))(B) = \mu(f^{-1}(B)) \quad B \in \Sigma_B \tag{6.38}$$

△

The pushforward measure on Y is simply the probability under $P(X)$ of the preimage of $f(Y)$. The detailed conditions on the pushforward measure are discussed in (specifically §3 and §9) of [50]. Further, a change of variables formula can be defined for the pushforward measure.

Definition 6.2.13. *Change of variables by pushforward measure:* By Theorem 3.6.1 in [50], with reference to given a pushforward measure as in Definition 6.2.12, the *change of variables by pushforward measure* of a function $g(b)$ on B by with measurable map $f : A \to B$ given probability measures $P(A)$, $P(B)$ on spaces A and B respectively is defined by:

$$\sum_{b \in B} g(b) P(a = f^{-1}(b)) = \sum_{a \in A} g(f(a)) P(a) \tag{6.39}$$

△

Equation (6.39) allows allows for a change of measure to be expressed without

reference to a function g (by taking $g = 1$) as:

$$\sum_{b \in B} P(a = f^{-1}(b)) = \sum_{a \in A} f(a) P(a) \tag{6.40}$$

6.2.4.3 Probabilistic forward and inverse problems

From Definition 6.2.13 of the change of variables by pushforward measure, both forward and inverse probabilistic problems (under a traditional residual minimisation solution framework) can be defined. For forward problems, a solution is defined as the output probability distribution $P(Y)$ given a residual function $H(y|x)$ and input probabilities $P(X)$. An inverse problem is defined as the input probability distribution $P(X)$ given a residual function $H(x|y)$ and input probabilities $P(Y)$. In both cases, the known measure on one space is pushed forward to the desired space. The change of variables formula can then be applied to find an expression for an estimate that can be calculated by a minimisation procedure.

Definition 6.2.14. *Probabilistic forward problem:* Given the input and output spaces X and Y, the input probability $P(X)$, the residual functional $H(y|x)$ and the measurable map:

$$u : X \longrightarrow Y \tag{6.41}$$

where u is a measurable map such that

$$u(x) := \underset{y \in Y}{\operatorname{argmin}}\ H(y = u(x)|x) \tag{6.42}$$

the *probabilistic forward problem solution* is defined as

$$P(Y)_{fwd} = \langle u(x) \rangle_{P(X)} = \sum_{y \in Y} y P(x = u^{-1}(y)) = \sum_{x \in X} u(x) P(x) \tag{6.43}$$

△

The condition that u be a measurable map is given in Definition 2.5.3. The intuitive understanding of what it means for u to be a measurable map is simply that $u^{-1}(B) \in X\ \forall B \in Y$, that is, the inverse under u of any subset in Y is in X.

Probabilistic inverse problems compute the expectation of a random variable

of the output space on the input space. To define the probabilistic inverse problems, consider the inverse of equation (6.42) and the definition of the inverse argmin from Definition 6.2.9.

$$X'(y) = \left[\underset{y' \in Y}{\text{argmin}}\ H(y|x)\right]^{-1}$$

The elements of the set $X'(y)$ is estimated in traditional inverse solution paradigms by minimisation of the induced residual $H(x|y) := \|x - u^{-1}(y)\|$, then as per the discussed in Section 6.2.4.1, the inverse $u^{-1}(y)$ can be estimated by an MLE of $H(x|y)$.

Definition 6.2.15. *Probabilistic inverse problem:* Given the input and output spaces X and Y, the output probability $P(Y)$, the residual functional $H(y|x)$ and the function:

$$u : X \longrightarrow Y \tag{6.44}$$

where u is a measurable map such that

$$u(x) := \underset{y \in Y}{\text{argmin}}\ H(y|x) \tag{6.45}$$

define the *induced inverse residual* given by the map:

$$H(x|y) : X \times Y \to \mathbb{R}^+ \tag{6.46}$$

as:

$$H(x|y) = \|x - u^{-1}(y)\| \tag{6.47}$$

the *inverse* as the set $X'(y) \in X$

$$X'(y) = \left\{X' \in \mathcal{P}(X) | x' \in X \wedge x \in X' = [\text{argmin}\ \ H(y|x)]^{-1}\right\} \tag{6.48}$$

Let $x'(y)$ be an element in the inverse such that:

$$u^{-1}(y) = x'(y) = \underset{x \in X}{\text{argmin}}\ H(x|y) \tag{6.49}$$

then the *probabilistic inverse problem solution* is defined as

$$P(X)_{inv} = \langle u^{-1}(y) \rangle_{P(Y)} = \sum_{x \in X} xP(y = u(x)) = \sum_{y \in Y} u^{-1}(y)P(y) \tag{6.50}$$

△

The relationship between the forward and inverse probabilistic problems is clear from equations (6.43) and (6.50). Both the forward and inverse problems evaluate a pushforward measure from one space to another based on the conditional distribution defined by a residual function.

The forward and inverse probability solutions are defined in terms of residual minimisation. As such these solutions are implicitly MLE's. Consider equation (6.26), which shows that the solution of forward problems found by residual minimisation are MLE's. Equation (6.33) and (6.24) indicate that minimum residual estimates for the inverse problem are also MLE's. Then the solution of forward and inverse problems can be expressed as:

$$P(Y)_{fwd} = \sum_{x \in X} y'(x)_{mle} P(x) \tag{6.51}$$

$$P(X)_{inv} = \sum_{y \in Y} x'(y)_{mle} P(y) \tag{6.52}$$

In Section 6.3.4 it is shown that these MLE's are a form of implicit Importance Sampling for estimating $P(Y)_{fwd}$ or $P(X)_{inv}$. As both the traditional forward and inverse techniques are essentially equivalent, the remaining discussion will focus mainly on the forward method with the understanding that the exposition is valid for both methods. By understanding probabilistic estimation in this way the theoretical and numerical problems associated with implicitly making these MLE's can be resolved. It will be demonstrated in Section 6.5 that the MLE assumptions required for traditional residual minimisation based solutions can increase the variance of estimated output probabilities and thus render a calculation useless. However, by understanding both perspectives, new light can be shed on existing techniques and how they may be modified to properly incorporate the Bayesian perspective in the future.

6.2.5 Deterministic problems and the β tolerance parameter

For completeness, this Section details how the solution of deterministic forward (and therefore inverse) problems can be bought into the Bayesian framework. This will also enable clarification of the β tolerance parameter from Definition 6.2.4. A deterministic problem can be understood as a problem for which

there is only a single input, x'. This can be interpreted directly as a Dirac delta prior such that $P(X) = \delta(x - x')$. Under a Bayesian framework, the deterministic case can be incorporated directly in a coherent manner, that is:

$$\begin{aligned} P(Y) &= \sum_{x \in X} P(Y|x)P(x) \\ P(Y) &= \sum_{x \in X} P(Y|x)\delta(x - x') \\ P(Y) &= P(Y|x') \end{aligned}$$

$P(Y) = P(Y|x')$ is defined by the output conditional probability, $P(Y)$ can be interpreted directly. Specifically:

$$-\ln P(Y) \propto \beta H(Y|x') \tag{6.53}$$

then, from equations (6.9,6.10) increasing β increases $P(Y)$. That is, increasing β increases the accessibility of higher values of $H(Y|x')$ and so increases the spread of $P(Y)$ over Y. β is in fact a Lagrange multiplier over the eigenvalues of the residual energy functional [83].

From an informatation-theoretic perspective, $-\ln P(y|x)$ defines the surprisal of $-\ln P(y|x)$ which is essentially the self-information of a particular observation from $H(y|x)$ [335, 83]. If the β is related to (that is, computations are regularised by) the machine precision of a computer, one would indeed expect that the effect of this coarse graining would be to increase the change to make observations from a broader region of Y.

6.3 Importance Sampling

6.3.1 Importance Sampling Estimates

A particular technique for estimating expectation values of quantities is *Importance Sampling* [320, 314]. In Section 6.4 it will be demonstrated that traditional Uncertainty Quantification is a form of Importance Sampling and so a full definition is supplied here. Importance sampling estimates $\langle \psi(a) \rangle_{P(A)}$ by introducing the likelihood ratio $w(a)$ (see Definition 6.3.1) and the proposal

distribution $Q(y)$ by calculating $\langle \psi(y) \rangle_{P(a)} = \langle \psi(a) w(a) \rangle_{Q(A)}$.

Definition 6.3.1. *Likelihood ratio:* Given the *nominal probability distribution* $P(A)$, and the *proposal distribution* (also termed the *importance distribution* [314]) the *likelihood ratio*, for $a \in A$, is:

$$w(a) = \frac{P(a)}{Q(a)} \tag{6.54}$$

△

Given the likelihood ratio, the Importance Sampling estimate can be found by expanding the definition of the expectation:

$$\begin{aligned}
\langle \psi(a) \rangle_{P(A)} &= \sum_{a \in A} \psi(a) P(a) \\
\langle \psi(a) \rangle_{P(A)} &= \sum_{a \in A} \psi(a) \frac{Q(a)}{Q(a)} P(a) \\
\langle \psi(a) \rangle_{P(A)} &= \sum_{a \in A} \psi(a) \frac{P(a)}{Q(a)} Q(a) \\
\langle \psi(a) \rangle_{P(A)} &= \sum_{a \in A} \psi(a) w(a) Q(a) \\
\langle \psi(a) \rangle_{P(A)} &= \langle \psi(a) w(a) \rangle_{Q(A)}
\end{aligned}$$

Under the condition that $q(x) > 0 \ \forall x \in X$ such that $\psi(x)P(x) \neq 0$ (see [320, 314] for a detailed derivation). Then the Importance Sampling estimate can be defined by:

Definition 6.3.2. *Importance Sampling estimate (ISE)*: Given:

$$\langle \psi(a) \rangle_{P(A)} = \sum_{a \in A} \psi(a) P(a) \tag{6.55}$$

and $Q(a)$ with $w(a) = \frac{P(a)}{Q(a)}$ such that condition that $q(x) > 0 \ \forall x \in X$ such that $\psi(x)P(x) \neq 0$, the *Importance Sampling estimate* of $\langle \psi(a) \rangle_{P(A)}$ is given by:

$$\langle \psi(a) \rangle_{P(A)} = \langle \psi(a) w(a) \rangle_{Q(A)} \tag{6.56}$$

△

6.3.2 Variance of Importance Sampling Estimates

Importance sampling is a particular *variance reduction technique* [320, 314]. By selecting different proposal distributions, the variance of an estimate can be modified [13]. The quality of one proposal distribution versus another can be defined in terms of the variance reduction achieved. The optimal proposal distribution is defined as that with zero variance. The optimal proposal distribution, however, depends on the function to be estimated. As the goal of this Chapter is to explore the effect of the implicit Importance Sampling used in traditional forward and inverse probability estimates, it is useful to state the variance of a proposal distribution.

From §4.3.1 in [320], the variance of an Importance Sampling estimate can be calculated directly in terms of the definition of variance, $\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, by:

$$\mathrm{Var}(\psi(a)w(a))_{Q(A)} = \langle(\psi(a)w(a))^2\rangle_{Q(A)} - \langle\psi(a)w(a)\rangle^2_{Q(A)} \tag{6.57}$$

$$\mathrm{Var}(\psi(a)w(a))_{Q(A)} = \sum_{a\in A}\frac{(\psi(a)P(a))^2}{Q(a)} - \langle\psi(a)w(a)\rangle^2_{Q(A)} \tag{6.58}$$

$$\mathrm{Var}(\psi(a)w(a))_{Q(A)} = \sum_{a\in A}\psi(a)^2 w(a)P(a) - \langle\psi(a)w(a)\rangle^2_{Q(A)} \tag{6.59}$$

$$\mathrm{Var}(\psi(a)w(a))_{Q(A)} = \langle\psi(a)^2 w(a)\rangle_{P(A)} - \langle\psi(a)w(a)\rangle^2_{Q(A)} \tag{6.60}$$

With reference to §4.3.1 in [320] and Theorem 3.3.4 in [314], the choice of $Q(a)$ that minimises the estimate variance, the *optimal proposal distribution* is given by:

$$Q^*(a) := \frac{|\psi(a)|P(a)}{\sum_{a'\in A}|\psi(a')|P(a')} \tag{6.61}$$

Unfortunately, it is impossible to evaluate the optimal proposal distribution exactly without knowing $\sum_{a'\in A}|\psi(a')|P(a')$ which is almost exactly the unknown expectation that is to be estimated by Importance Sampling in the first place. A large number of techniques exist for approximating the optimal sampling distribution for example using approximate normalisation (§3.2.2 in [314] and the powerful cross entropy method [319].

6.3.3 Unnormalised Importance Sampling

A crucial aspect of Importance Sampling is that the proposal distribution does not need to be normalised to make numerical estimates [320, 314]. By setting up the problem correctly, the normalisation constants of the proposal distribution can be essentially divided away. This is a well known, but very useful feature of Importance Sampling as normalisation constants (or partition functions) can be very difficult or impossible to feasibly calculate analytically [233]. As the unnormalised estimate is a key part in the main result of this Chapter (that traditional residual minimisation techniques for Uncertainty Quantification are an implicit form of Importance Sampling) it is useful to provide the details of the derivation of the unnormalised estimation technique.

To derive the unnormalised estimate, begin with the unnormalised distributions $\tilde{P}(A)$ and $\tilde{Q}(A)$. Define the following:

$$P(A) := \frac{\tilde{P}(X)}{Z_P} \qquad Z_P := \sum_{a \in A} \tilde{P}(a) \tag{6.62}$$

$$Q(A) := \frac{\tilde{Q}(X)}{Z_Q} \qquad Z_P := \sum_{a \in A} \tilde{Q}(a) \tag{6.63}$$

$$\tilde{w}(a) := \frac{\tilde{P}(a)}{\tilde{Q}(a)} \tag{6.64}$$

then expand the required expectation value, $\langle f(a) \rangle_{P(A)}$, in terms of $P(A)$ and $Q(A)$:

$$\begin{aligned}
\langle f(a) \rangle_{P(A)} &= \sum_{a \in A} f(a) P(a) \\
\langle f(a) \rangle_{P(A)} &= \sum_{a \in A} f(a) \frac{P(a)}{Q(a)} Q(a) \\
\langle f(a) \rangle_{P(A)} &= \sum_{a \in A} f(a) \frac{\tilde{P}(a)}{Z_P} \frac{Z_Q}{\tilde{Q}(a)} Q(a) \\
\langle f(a) \rangle_{P(A)} &= \sum_{a \in A} f(a) \tilde{w}(a) \frac{Z_Q}{Z_P} Q(a) \\
\langle f(a) \rangle_{P(A)} &= \frac{Z_Q}{Z_P} \sum_{a \in A} f(a) \tilde{w}(a) Q(a)
\end{aligned}$$

Next, expand the ratio of the unknown normalising constants:

$$\frac{Z_P}{Z_Q} = \frac{1}{Z_Q} \sum_{a \in A} \tilde{P}(A) \tag{6.65}$$

Note that, by reorganising equation (6.63):

$$Z_Q = \frac{\tilde{Q}(A)}{Q(A)}$$

so equation (6.65) becomes:

$$\frac{Z_P}{Z_Q} = \sum_{a \in A} \frac{Q(a)}{\tilde{Q}(a)} \tilde{P}(a)$$
$$\frac{Z_P}{Z_Q} = \sum_{a \in A} \tilde{w}(a) Q(a)$$

which can be written expressed as the useful equation:

$$\frac{Z_P}{Z_Q} = \langle \tilde{w}(a) \rangle_{Q(A)} \tag{6.66}$$

Then, the unnormalised Importance Sampling estimate can be written:

$$\langle f(a) \rangle_{P(A)} = \frac{\langle f(a) \tilde{w}(a) \rangle_{Q(A)}}{\langle \tilde{w}(a) \rangle_{Q(A)}} \tag{6.67}$$

The expectations in both the numerator and denominator in equation (6.67) can be estimated numerically by Monte Carlo Simulation, defined in Section 6.3.4.

6.3.3.1 Variance of unnormalised Importance Sampling estimates

The variance of an unnormalised Importance Sampling estimate can be calculated in essentially the same way as for the normalised case. The unnormalised variance estimate will be a key component for understanding the failure of traditional methods on certain problems and so is given here in detail. From the

definition of variance:

$$\mathrm{Var}(\psi(a)w(a))_{Q(A)} = \sum_{a\in A}\psi(a)^2 w(a)^2 Q(a) - \left(\sum_{a\in A}\psi(a)w(a)Q(a)\right)^2$$

$$\mathrm{Var}(\psi(a)w(a))_{Q(A)} = \sum_{a\in A}\psi(a)^2\left(\frac{Z_Q}{Z_P}\right)^2\tilde{w}(a)^2 Q(a) - \left(\sum_{a\in A}\psi(a)\left(\frac{Z_Q}{Z_P}\right)Q(a)\right)^2$$

$$\mathrm{Var}(\psi(a)w(a))_{Q(A)} = \left(\frac{Z_Q}{Z_P}\right)^2\langle\psi(a)^2\tilde{w}(a)^2\rangle_{Q(A)} - \left(\frac{Z_Q}{Z_P}\right)^2\left(\langle\psi(a)\tilde{w}(a)\rangle_{Q(A)}\right)^2$$

from equation (6.66), the ratio of the partition function terms can be written:

$$\mathrm{Var}(\psi(a)w(a))_{Q(A)} = \frac{\langle\psi(a)^2\tilde{w}(a)^2\rangle_{Q(A)} - \left(\langle\psi(a)\tilde{w}(a)\rangle_{Q(A)}\right)^2}{\langle\tilde{w}(a)^2\rangle_{Q(A)}} \tag{6.68}$$

The above derivation has yielded the variance of an Importance Sampling estimate using an unnormalised proposal distribution. This can be written in a shorthand notation that should be clear from the context:

$$\mathrm{Var}(\psi(a))_{P(A)} = \frac{\langle\psi^2\tilde{w}\rangle - \langle\psi\tilde{w}\rangle^2}{\langle\tilde{w}\rangle^2} \tag{6.69}$$

By inspection of equation (6.69), when estimating the variance terms numerically, difficulties may when $\langle\tilde{w}(a)\rangle_{Q(A)}$ is very small (and so $\langle\tilde{w}(a)\rangle^2_{Q(A)}$ is even smaller) when ψ is not. In this case, catastrophic cancellation (see §1.7 in [178]) of the variance reducing negative term in equation (6.69) will render this term unable to reduce the estimated variance. This will occur when $\langle\tilde{w}(a)\rangle$ is far from unity. These derivations will be useful when considering the further developments in Section 6.4.

6.3.4 Monte Carlo Simulation

To numerically estimate expectation values, standard *Monte Carlo Simulation*, defined in detail in [118, 314], will be used.

Definition 6.3.3. *Monte Carlo Simulation (MCS):* Given spaces A and B, a function $f : A \to B$ and probability distribution $P(A)$, the *Monte Carlo*

estimate of the expectation of $f(a)$ over $P(A)$ is:

$$\langle f(a)\rangle_{P(A)} \approx \hat{\mu}(f(a))_{P(A)} := \frac{1}{N_A}\sum_{i=i}^{N_A} f(A_i) \quad A_i \sim P(A)$$

where $A_i \sim P(A)$ denotes *sampling* from $P(A)$ and N_A is the total number of samples drawn from $P(A)$. △

Further, by the Central Limit Theorem it is a standard result (see [314]) that given $\hat{\mu}(f(a))_{P(A)}$ and the variance of the samples $\text{Var}\, f(A_i) = \hat{\sigma}^2(f(a))_{P(A)}$, the mean and variance of the MCS estimate is normally distributed:

$$\hat{\mu}(f(a))_{P(A)} \sim \mathcal{N}\left(\hat{\mu}(f(a))_{P(A)}, \frac{\hat{\sigma}^2(f(a))_{P(A)}}{N_A}\right) \tag{6.70}$$

so that the standard deviation of the MCS estimate of $\hat{\mu}(f(a))_{P(A)}$ decreases like $N_A^{-\frac{1}{2}}$.

6.4 Traditional Uncertainty Quantification is a form of Importance Sampling

This Section presents the main result of this Chapter, that traditional Uncertainty Quantification can be interpreted as Importance Sampling using MLE's. The overall structure of this demonstration is as follows. First, a QoI estimation on the output space is framed as an estimate on the joint input-output probability space. Next, Importance Sampling is introduced to this estimate. By a particular choice of proposal distribution, the Importance Sampling estimate of the QoI is shown to be almost equivalent to the standard estimates as defined by equations (6.51) and (6.52). In particular, it is shown that an additional likelihood term prevents the identification. This likelihood term is the probability of the output given the input, that is the likelihood of the input given the output. By assuming this probability is unity, the traditional estimate is recovered. The significance of the assumption that this likelihood term can be ignored is discussed in Section 6.4.4 and explored numerically in Section 6.5. The variance of the implicit traditional method proposal distribution is detailed. By understanding both the output-input likelihood term and the variance of Importance Sampling estimates, the relative success and

failure of traditional approaches to Uncertainty Quantification for different QoI estimation problems can be understood clearly.

6.4.1 Equating the Bayesian and traditional perspectives

The goal of this Section (and primary contribution of this Chapter) is to demonstrate the (almost) equivalence of the Bayesian QoI estimates by Importance Sampling and the traditional forward and inverse QoI estimates based on residual minimisation. From the discussion in Section 6.2.4, the forward and inverse problems can be framed essentially identically as pushforward measures from one space to another. Then, to demonstrate that the traditional techniques are an implicit form of Importance Sampling it will be sufficient to show in this Section that, for the forward case, the following is satisfied:

$$\langle \psi(y) \rangle_{P(Y)} \stackrel{?}{=} \langle \psi(u(x)) \rangle_{P(X)} \tag{6.71}$$

where ψ is a QoI function, $\langle \psi(y) \rangle_{P(Y)}$ represents the Bayesian QoI estimate and $\langle \psi(u(x)) \rangle_{P(X)}$ represents the forward QoI estimate. It will be shown that equation (6.71) can not be satisfied exactly. However, the traditional approach and the Bayesian estimates are approximately equal, but that the lack of exact equality of these estimates is precisely the term required to render the traditional forward and inverse methods coherent when multiple minima of the residual function are present.

6.4.2 Expansion of Bayesian QoI on the joint space

The Bayesian QoI estimate can be expressed as a QoI on the joint space. The typical case that $\psi(y)$ is independent of x is shown. This is without significant loss of generality from the case that $\psi(x, y)$ (the QoI function is a

joint probability on the input and output). For $\psi(y)$:

$$\begin{aligned}
\langle\psi(y)\rangle_{P(Y)} &= \sum_{y\in Y}\psi(y)P(y) \\
\langle\psi(y)\rangle_{P(Y)} &= \sum_{y\in Y}\psi(y)\sum_{x\in X}P(y|x)P(x) \\
\langle\psi(y)\rangle_{P(Y)} &= \sum_{y\in Y}\psi(y)\sum_{x\in X}P(x,y) \\
\langle\psi(y)\rangle_{P(Y)} &= \sum_{y\in Y}\sum_{x\in X}\psi(y)P(x,y) \\
\langle\psi(y)\rangle_{P(X,Y)} &= \sum_{x\in X}\sum_{y\in Y}\psi(y)P(x,y)
\end{aligned}$$

then

$$\langle\psi(y)\rangle_{P(Y)} = \langle\psi(y)\rangle_{P(X,Y)} \tag{6.72}$$

Introducing an Importance Sampling estimate with proposal distribution $Q(x,y)$ and expanding yields:

$$\langle\psi(y)\rangle_{P(X,Y)} = \langle\psi(y)w(x,y)\rangle_{Q(X,Y)} \tag{6.73}$$

$$\langle\psi(y)w(x,y)\rangle_{Q(X,Y)} = \sum_{x\in X}\sum_{y\in Y}\psi(y)\frac{P(x,y)}{Q(x,y)}Q(x,y) \tag{6.74}$$

or, if an unnormalised proposal distribution $\tilde{Q}(x,y)$ is used:

$$\langle\psi(y)\rangle_{P(X,Y)} = \frac{\langle\psi(y)\tilde{w}(x,y)\rangle_{Q(X,Y)}}{\langle\tilde{w}(x,y)\rangle_{Q(X,Y)}} \tag{6.75}$$

$$\langle\psi(y)\rangle_{P(X,Y)} = \frac{\sum_{x\in X}\sum_{y\in Y}\psi(y)\frac{\tilde{P}(x,y)}{\tilde{Q}(x,y)}Q(x,y)}{\sum_{x\in X}\sum_{y\in Y}\frac{\tilde{P}(x,y)}{\tilde{Q}(x,y)}Q(x,y)} \tag{6.76}$$

6.4.3 The forward problem case

6.4.3.1 The required proposal distribution

First, the forward and Bayesian QoI estimates will be related by Importance Sampling. $Q(x,y)$ must be selected so as to recover the forward QoI estimate from the Bayesian case. To this end, let $\tilde{Q}(x,y)$ be given by an unnormalised probability distribution that samples the same y that would be found by $u(x)$

with x selected with probability $P(x)$:

$$\tilde{Q}(x,y) = \delta\left(y - \underset{y \in Y}{\operatorname{argmax}}\ P(y|x)\right) P(x) \tag{6.77}$$

that is:

$$\begin{aligned}
\tilde{Q}(x,y) &= \tilde{Q}(y|x)\tilde{Q}(x) \\
\tilde{Q}(y|x) &= \delta\left(y - \underset{y \in Y}{\operatorname{argmax}}\ P(y|x)\right) \\
\tilde{Q}(x) &= P(x)
\end{aligned}$$

To be a valid proposal distribution, it is a requirement that $\tilde{Q}(x,y) > 0$ if $\psi(y)P(x,y) \neq 0$ and that $Z_{Q(X,Y)} > 0$. These conditions are proven in Theorems 6.6.1 and 6.6.2 presented in the Chapter Appendix.

Note that, from equation (6.120), although $Q(X,Y)$ is in fact normalised over the joint space $X \times Y$, it will not be normalised when used as the proposal distribution for emulating the traditional forward probability estimate by the Bayesian approach. This will be demonstrated as part of the developments in the following section, justifying the use of the unnormalised Importance Sampling technique to relate the Bayesian and traditional forward probabilistic approaches.

6.4.3.2 Expanding the Importance Sampling estimate

For convenience, with reference to equation (6.23), define:

$$y'(x)_{mle} := \underset{y \in Y}{\operatorname{argmax}}\ P(y|x) \tag{6.78}$$

Using conditional expectations [224]:

$$\frac{P(x,y)}{P(x)} = P(y|x) \tag{6.79}$$

Then, expanding the required proposal distribution equation (6.77) into equa-

tion (6.76) yields, for the numerator:

$$\langle \psi(y)\tilde{w}(x,y)\rangle_{Q(X,Y)} = \sum_{x\in X}\sum_{y\in Y}\psi(y)\frac{\tilde{P}(x,y)\left[\delta\left(y-y'(x)_{mle}\right)P(x)\right]}{\left[\delta\left(y-y'(x)_{mle}\right)P(x)\right]}$$

$$\langle \psi(y)\tilde{w}(x,y)\rangle_{Q(X,Y)} = \sum_{x\in X}\sum_{y\in Y}\psi(y)\frac{P(x,y)Z_{P(X,Y)}\left[\delta\left(y-y'(x)_{mle}\right)P(x)\right]}{\left[\delta\left(y-y'(x)_{mle}\right)P(x)\right]}$$

$$\langle \psi(y)\tilde{w}(x,y)\rangle_{Q(X,Y)} = \sum_{x\in X}\sum_{y\in Y}\psi(y)\frac{P(y|x)Z_{P(X,Y)}\left[\delta\left(y-y'(x)_{mle}\right)P(x)\right]}{\left[\delta\left(y-y'(x)_{mle}\right)\right]}$$

$$\langle \psi(y)\tilde{w}(x,y)\rangle_{Q(X,Y)} = \sum_{x\in X}\psi(y'(x)_{mle})P(y=y'(x)_{mle}|x)Z_{P(X,Y)}P(x)$$

$$\langle \psi(y)\tilde{w}(x,y)\rangle_{Q(X,Y)} = \langle\psi(y)P(y=y'(x)_{mle}|x)Z_{P(X,Y)}\rangle_{P(X)}$$

$$\langle \psi(y)\tilde{w}(x,y)\rangle_{Q(X,Y)} = Z_{P(X,Y)}\langle\psi(y)P(y=y'(x)_{mle}|x)\rangle_{P(X)}$$

and for the denominator, following the essentially identical expansion used for the numerator:

$$\langle \tilde{w}(x,y)\rangle_{Q(X,Y)} = \sum_{x\in X}\sum_{y\in Y}\psi(y)\frac{P(y|x)Z_{P(X,Y)}\left[\delta\left(y-y'(x)_{mle}\right)P(x)\right]}{\left[\delta\left(y-y'(x)_{mle}\right)\right]}$$

$$\langle \tilde{w}(x,y)\rangle_{Q(X,Y)} = \sum_{x\in X}P(y=y'(x)_{mle}|x)Z_{P(X,Y)}P(x)$$

$$\langle \tilde{w}(x,y)\rangle_{Q(X,Y)} = Z_{P(X,Y)}\langle P(y=y'(x)_{mle}|x)\rangle_{P(X)}$$

Then, combining the above expressions for the numerator and denominator of equation (6.76):

$$\langle\psi(y)\rangle_{P(X,Y)} = \frac{\langle \psi(y)\tilde{w}(x,y)\rangle_{Q(X,Y)}}{\langle \tilde{w}(x,y)\rangle_{Q(X,Y)}}$$

$$\langle\psi(y)\rangle_{P(X,Y)} = \frac{Z_{P(X,Y)}\langle\psi(y)P(y=y'(x)_{mle}|x)\rangle_{P(X)}}{Z_{P(X,Y)}\langle P(y=y'(x)_{mle}|x)\rangle_{P(X)}}$$

$$\langle\psi(y)\rangle_{P(X,Y)} = \frac{\langle\psi(y)P(y=y'(x)_{mle}|x)\rangle_{P(X)}}{\langle P(y=y'(x)_{mle}|x)\rangle_{P(X)}}$$

Finally, the crucial result for the Bayesian QoI forward Importance Sampling estimate using the given $\tilde{Q}(x,y)$ is:

$$\langle\psi(y)\rangle_{P(X,Y)} = \frac{\langle\psi(y)P(y=y'(x)_{mle}|x)\rangle_{P(X)}}{\langle P(y=y'(x)_{mle}|x)\rangle_{P(X)}} \tag{6.80}$$

This can be expressed in a useful shorthand notation, whose meaning should be clear from the context, where $\mathcal{L} := \tilde{P}(y'|x)$ (the likelihood calculated by the unnormalised $\tilde{P}(y'|x)$):

$$\langle \psi(y) \rangle_{P(X,Y)} = \frac{\langle \psi \mathcal{L} \rangle}{\langle \mathcal{L} \rangle} \tag{6.81}$$

6.4.4 Comparison of Bayesian Importance Sampling and traditional forward QoI estimates

The forward QoI estimate can be expressed, from equation (6.51), as:

$$\langle \psi(u(x)) \rangle_{P(X)} = \sum_{x \in X} \psi(y'(x)_{mle}) P(x) \tag{6.82}$$

Comparison of the Bayesian estimate in equation (6.80) and the forward QoI estimate in equation (6.82) is revealing. The Bayesian estimate with the appropriate proposal distribution is precisely the expectation of $\psi(y)P(y = y'(x)_{mle}|x)$ over $P(X)$, normalised by $P(y = y'(x)_{mle}|x)$. The traditional forward estimate ignores the $P(y = y'(x)_{mle}|x)$ term. This term is precisely the likelihood, with reference to equation (6.18), to have sampled the $y'(x)_{mle}$ given the Gibbs distribution defined by $H(y|x)$:

$$P(y = y'(x)_{mle}|x) = \mathcal{L}(y'(x)_{mle}; x, y) \tag{6.83}$$

Further, by comparison with the definition of unnormalised Importance Sampling in Section 6.3.3, $\tilde{w}(x, y) = P(y = y'(x)_{mle}|x)$, so $\mathcal{L}(y'(x)_{mle}; x, y)$ is exactly the importance weight assigned to the sample $y'(x)_{mle}$. The implicit assumption made by traditional forward approaches is then made clear. Forward solutions based on minimum residual estimates assume that $P(y = y'(x)_{mle}|x) = 1$. It is this assumption that allows for the likelihood term $\mathcal{L}(y'(x)_{mle}; x, y)$ to be neglected without severe penalty for many problems. For multimodal residual surfaces, this assumption is, however, clearly invalid, especially if a local (rather than global) energy minimum is found by an MLE. The higher the energy of a local minimum, the smaller the value of $\mathcal{L}(y'(x)_{mle}; x, y)$. Additionally, if $H(y|x)$ has degenerate ground states (that is, a global minimal value that is the same in multiple locations within the output space), then the likelihood of each of these degenerate ground states

will be less than unity. Traditional forward approaches fail in these situations not least because the estimates have not been correctly normalised.

Why, then, has the traditional forward approach been successful for a large number of problems, for example in [116, 118, 141, 351]. If the energy surface defined by $H(y|x)$ has a single minimum and a high solution tolerance is used (i.e. β is very large) then $\mathcal{L}(y'(x)_{mle}; x, y)$ approaches 1 and the error caused by ignoring $\mathcal{L}(y'(x)_{mle}; x, y)$ is small. The most common $H(y|x)$ encountered in energy minimisation problems for partial differential equations is the quadratic energy surface as this appears in the solution of the Laplace equation (equivalently the Poisson equation) [198]. The very common heat, wave and advection-diffusion equations all feature a Laplacian term meaning that the energy surface is frequently quadratic for parts or all of $H(y|x)$. Similarly, the standard harmonic oscillator which is a central part of many dynamical problems [260] also features a quadratic Hamiltonian. The unimodality of the energy surface for all of these problems lessens the error caused by the implicit assumption that $\mathcal{L}(y'(x)_{mle}; x, y) = 1$ for high tolerance problems.

6.4.4.1 Variance of the forward estimate

This Section has detailed how the traditional forward method for making QoI estimates is an implicit form of Importance Sampling with the additional assumption that $\mathcal{L}(y'(x)_{mle}; x, y) = 1$. The proposal distribution, $Q(x, y)$, implicit in the forward method is given by equation (6.77). The quality of a proposal distribution can be defined in terms of the Importance Sampling variance reduction achieved by the use of one proposal distribution instead of another as discussed in Section 6.3.2. To further analyse the effect of the implicit $Q(x, y)$ used by the forward probability estimation technique, consider that the variance of this estimate can be found using equation (6.68). First, for convenience in this section write let:

$$y' := y'(x)_{mle} \tag{6.84}$$

then the variance of the estimate is given by:

$$\mathrm{Var}(\psi(y))_{P(X,Y)} = \frac{\langle \psi(y)^2 \tilde{w}(x,y) \rangle_{Q(X,Y)} - \langle \psi(y) \tilde{w}(x,y) \rangle^2_{Q(X,Y)}}{\langle \tilde{w}(x,y) \rangle^2_{Q(X,Y)}}$$
$$\mathrm{Var}(\psi(y))_{P(X,Y)} = \frac{\langle \psi(y')^2 P(y'|x) \rangle_{P(X)} - \langle \psi(y') P(y'|x) \rangle^2_{P(X)}}{\langle P(y'|x) \rangle^2_{P(X)}}$$

As in equation (6.81) for the expectation of the Bayesian forward estimate, the variance can written in shorthand notation where $\mathcal{L} := \tilde{P}(y'|x)$ (the likelihood calculated by the unnormalised $\tilde{P}(y'|x)$):

$$\mathrm{Var}(\psi(y))_{P(X,Y)} = \frac{\langle \psi^2 \mathcal{L} \rangle - \langle \psi \mathcal{L} \rangle^2}{\langle \mathcal{L} \rangle^2} \tag{6.85}$$

Independent of the relationship with $\psi(y)$, the crucial part in the above equation is that:

$$\mathrm{Var}(\psi(y))_{P(X,Y)} \propto \frac{1}{\langle P(y = y'(x)_{mle}|x) \rangle^2_{P(X)}} \tag{6.86}$$

Equation (6.86) essentially says that if the true likelihood of the MLE for y is small, then the variance of the estimate will likely be very large, amplifying any lack of proportionality in the numerator of the full expression for the variance. The likelihood term will be large for multimodal distributions or MLE's that are local optima far from the true minimum residual.

6.4.4.2 Numerically estimating the Importance Sampling estimate

From equation (6.70), as discussed in Section 6.3.4, when expectations are evaluated by Monte Carlo Simulation the calculated value and error of the estimate are described by the mean and variance of normal distribution. Specifically, after N simulations the estimated value of f will be distributed as $\langle f \rangle \approx \mathcal{N}(\mu, \sigma^2)$ and the estimated mean value of like $\langle f \rangle \approx \mathcal{N}(\mu, \frac{\sigma^2}{N})$. The variance of the Bayesian Importance Sampling estimate, equation (6.86), is a more accurate form of the residual minimisation forward estimate. These forward estimates may be calculated by MCS and as such the error in the estimate (and therefore confidence intervals) can be found directly. However, based on the previous discussion, the correct form of the residual minimisation forward estimate is given by the Bayesian estimate in equation (6.80).

Expansion of the variance of this estimate yielded equation (6.85). In MCS, each of the terms in equation (6.85) will be evaluated by separate MCS estimates. Each of these individual estimates will have a normal distribution. The correct Bayesian form of the forward problem variance indicates then that the true variance of the MCS estimate made using residual minimisation cannot be described by just a normal distribution. This is because the product and quotient of several normal random variables is not necessarily a normal distribution [157].

Let $\overline{\langle f \rangle}$ denote the MCS estimate of the expectation of f. First, for convenience define the mean and variance of an MCS estimate made using N simulations by:

$$\langle f \rangle \approx \overline{\langle f \rangle} = \mathcal{N}\left(\hat{\mu}(f), \frac{\hat{\sigma}^2(f)}{N}\right)$$

Then, the true MCS estimate distribution of the correct Bayesian version of the forward problem is then:

$$\mathrm{Var}(\psi(y))_{P(X,Y)} \approx \frac{\overline{\langle \psi^2 \mathcal{L} \rangle} - \overline{\langle \psi \mathcal{L} \rangle}^2}{\overline{\langle \mathcal{L} \rangle}^2} \tag{6.87}$$

$$\mathrm{Var}(\psi(y))_{P(X,Y)} \approx \frac{\left[\mathcal{N}\left(\hat{\mu}(\psi^2\mathcal{L}), \frac{\hat{\sigma}^2(\psi^2\mathcal{L})}{N}\right)\right] - \left[\mathcal{N}\left(\hat{\mu}(\psi\mathcal{L}), \frac{\hat{\sigma}^2(\psi\mathcal{L})}{N}\right)\right]^2}{\left[\mathcal{N}\left(\hat{\mu}(\mathcal{L}), \frac{\hat{\sigma}^2(\mathcal{L})}{N}\right)\right]^2} \tag{6.88}$$

Equation (6.88) gives the MCS estimate of the true variance of an estimate made by a traditional, residual minimisation MCS estimate. By considering the Bayesian perspective of solving equations, multimodality of the residual function can be correctly considered. Unfortunately, as the product and quotient of normal distributions is not a normal distribution, estimating this variance requires more care. If only an estimate of the variance is required, then equation (6.69) can be estimated by the mean values calculated using MCS directly. The fully rigorous calculation of confidence intervals would require, however, integration over the densities defined by equation (6.88) as the estimated variance distribution will not be of a simple form.

6.4.5 The inverse problem case

6.4.5.1 The required proposal distribution

Inverse probability problems based on residual minimisation can be understood in a virtually identical manner to the forward case, as shown in Section 6.2.4. Again, a Bayesian QoI estimate can be made using Importance Sampling with $Q(x, y)$ designed to recover the implicit Importance Sampling used in the traditional inverse problem approach.

From Definition 6.2.15 and equation (6.71), the inverse QoI estimate is

$$\langle \phi(u^{-1}(y)) \rangle_{P(Y)} \tag{6.89}$$

If each inverse is estimated by minimising the residual $H(x|y) = \|x - x'(y)_{mle}\|$, then the inverse is a set $X'(y)_{mle} = H(x|y)$. However if, as is the case in regularised inverse problem solutions [178, 355], it is assumed that the inverse can be approximated by being sufficiently close to a particular value then the traditional inverse problem solution can be interpreted as

$$\langle \phi(u^{-1}(y)) \rangle_{P(Y)} = \sum_{y \in Y} \phi(x'(y)_{mle}) P(y) \tag{6.90}$$

In the Bayesian interpretation, the inverse estimates $x'_{mle}(y)$ can be interpreted as being sampled from $u^{-1}(Y)$ with probability distribution defined by Gibbs measure with the induced inverse residual as the energy function (see Definition 6.2.15):

$$H(x|y) = \|x - x'(y)_{mle}\| \tag{6.91}$$

Then, the required Importance Sampling proposal distribution to recover the inverse case from the Bayesian interpretation is

$$\tilde{Q}(x, y) = \delta \left(x - \underset{x \in X}{\operatorname{argmin}}\ H(x|y) \right) P(y) \tag{6.92}$$

By exact analogy with Theorems 6.6.1 and 6.6.2 in the Chapter Appendix for the forward case, this is a valid proposal distribution as $Q(x, y) \geq 0$ for all $\psi(y)P(x, y) \neq 0$ and $Z_Q \geq 0$. Also by exact analogy with the forward case,

the Bayesian Importance Sampling estimate using $Q(x, y)$ is given by:

$$\langle \psi(y) \rangle_{P(X,Y)} = \frac{\langle \psi(y) P(x = x'(y)_{mle}|y) \rangle_{P(Y)}}{\langle P(x = x'(y)_{mle}|y) \rangle_{P(Y)}} \tag{6.93}$$

As in the forward case, the Bayesian interpretation renders obvious the implicit assumption in traditional inverse approaches that the likelihood of drawing a particular inverse solution can be ignored (or calculated automatically). The effect of ignoring this normalisation on the variance of a QoI estimate is essentially the same as in the forward case discussed in Section 6.4.4.1.

6.4.6 Discussion

It was demonstrated in this Section that both the traditional forward and inverse approaches to Uncertainty Quantification can be recast Bayesian terms. The benefit of the Bayesian perspective is that it makes the assumptions implicit in the traditional approaches transparent. In particular, it was demonstrated that the traditional approaches can be viewed as a type of Importance Sampling that uses Maximum Likelihood Estimates as a proposal distribution. The variance of these estimates was explored in detail. By understanding the variance in the Importance Sampling estimates that result from the implicit proposal distributions of traditional forward and inverse techniques, it is possible to better understand the types of problems for which the traditional approaches have been or have not been successful. In particular, by understanding the effect of the MLE likelihood on sample variances, lower variance estimators can be constructed. Numerical experiments supporting these developments are presented in the next Section.

6.5 Numerical Experiments

This Chapter has so far explored the theoretical links between the traditional and Bayesian approaches to Uncertainty Quantification for estimating Quantities of Interest. In this Section, several numerical experiments are presented to test the theoretical findings. As this Chapter is exploring the effect of residual minimisation only on the solution of equations, rather than on fitted probability density estimates, all of the numerical problems presented test threshold

indicator QoI's on the output space. Threshold indicator problems are common in reliability analysis [118, 356, 357, 21, 16] and are, as such, useful to study. The correct probability estimates (assuming a uniform prior) are given by the beta distribution (§8 in [224]). The single valued threshold probability estimates for all numerical problems presented in this Section are then, in a sense, MLE's from the beta distribution.

The first problem investigates the effect of the implicit proposal distribution for forward estimates on far from mean threshold probability estimates. The second problem demonstrates how global minima far from the threshold to be estimated can destroy the accuracy traditional forward estimates. The Bayesian viewpoint is then used to suggest an improved sampling method. The final problem tests the effect of multimodality. For all problems, directly considering the joint space probability density rather than MLE's shown to provide better estimates of the true threshold probability.

For each problem, an understanding of the variance of Importance Sampling is used to generate improved proposal distributions. In the first problem, a better forward proposal distribution is tested which uses still uses residual minimisation. It is shown that this method offers some, but not drastic improvements. In the second problem, a proposal distribution on the joint space is used which avoids taking MLE's altogether. This is shown to perform well. Further, the choice of proposal distribution could have been found with knowledge only of the conditional residual energy surface, $H(y|x)$. This is knowledge which would be available when solving a real problem. Finally, in the third problem it is demonstrated that the effect of more complicated multimodality of $H(y|x)$ can be mitigated by Markov Chain Monte Carlo sampling on the joint input-output space.

6.5.1 Unimodal residual function - multivariate Gaussian threshold estimation

In this problem, a bivariate normal is used to represent a unimodal joint input-output distribution. A series of threshold probability estimates on the output space that move away from the mean are calculated. The MLE based forward approach and Bayesian correction to this method are compared. Further, it is demonstrated that by directly sampling from the joint probability space much

better Quantity of Interest estimates can be obtained. From the specified joint distribution, the residual $H(y|x)$ is found analytically. This allows for the numerical probability evaluations to be compared with direct integration of the known solution. Note that for a real problem, the probability $P(X, Y)$ is not known a priori because, if it were, the problem would be solved already. Thus, the direct integration approach would not typically be available for Uncertainty Quantification. The problem presented will, however, demonstrate the effect of including the likelihood terms in the forward method implicit Importance Sampling estimate. The effect of the Bayesian correction to the forward probability estimate is small because of the unimodality of the global minimum. However, by considering the Bayesian perspective the utility using of the joint input-output distribution for sampling becomes apparent.

6.5.1.1 Problem specification

Consider the k dimensional multivariate normal distribution with mean vector μ and correlation matrix Σ:

$$f(a, b|\mu, \sigma) = (2\pi)^{-\frac{k}{2}}|\Sigma|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{6.94}$$

Specifically, consider the bivariate case with mean μ and correlation matrix Σ with entries:

$$\mu = \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_A^2 & \rho\sigma_A\sigma_B \\ \rho\sigma_A\sigma_B & \sigma_B^2 \end{bmatrix} \tag{6.95}$$

For this problem, an unnormalised bivariate Gaussian will be used to represent the joint distribution of the input and output spaces X and Y. For $\rho > 0.0$, $P(X, Y) = P(Y|X)P(X)$ can be used to find an unnormalised analytical expression for $\beta H(Y|X) \propto -\ln P(Y|X)$. First, consider the conditional bivariate Gaussian distribution [375]:

$$f(y|x, \mu, \sigma) = \mathcal{N}\left(\mu_Y + \frac{\sigma_Y}{\sigma_X}\rho(x - \mu_X), (1 - \rho^2)\sigma_Y^2\right) \tag{6.96}$$

For this problem, let $\rho = 0.5$, $\mu_X = \mu_Y = 0$ and $\sigma_X = \sigma_Y = 1.0$ so that

$$P(Y|x) = \mathcal{N}\left(\frac{x}{2}, \frac{3}{4}\right) \tag{6.97}$$

then $H(y|x)$ is:

$$H(y|x) = \left(\frac{2}{3}\left[y^2 - xy + \frac{x^2}{4}\right]\right) \tag{6.98}$$

The problem is to evaluate, given the input distribution $P(X) = \mathcal{N}(0,1)$, the threshold indicator Quantity of Interest $\psi_\gamma(y)$ on the output space:

$$\psi_\gamma(y) = \chi_{y \geq \gamma}(y) \tag{6.99}$$

for:

$$\gamma = \{\gamma_0, \gamma_1, \gamma_2\} = \{0, 1, 2\} \tag{6.100}$$

A representation of the threshold density estimation problem is shown in Figure 6.1.

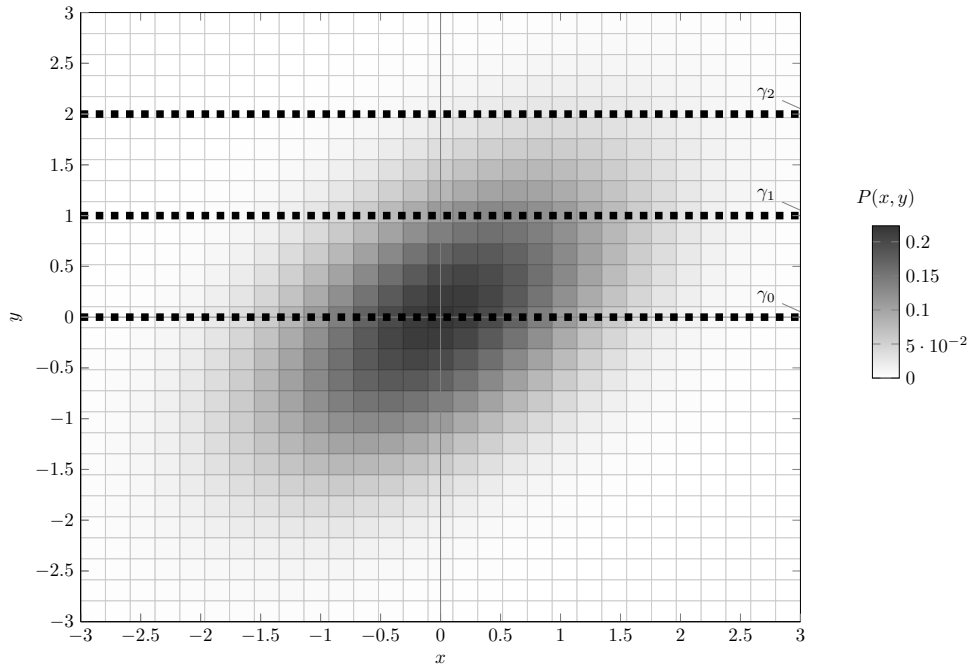


Figure 6.1: Unimodal threshold probability estimation problem overview: find $P(y \geq \gamma_i) = ?$.

6.5.1.2 Comparison of Bayesian Importance Sampling and forward QoI estimates

To estimate $\psi_\gamma(y)$, the traditional forward QoI estimate was compared with the Bayesian version of this estimate. The residual minimising forward esti-

mate and Bayesian estimate are given by equations (6.82) and (6.80) respectively as:

$$\langle \psi_\gamma(u(x)) \rangle_{P(X)} = \sum_{x \in X} \psi_\gamma(y'(x)_{mle}) P(x)$$

$$\langle \psi_\gamma(y) \rangle_{P(X,Y)} = \frac{\langle \psi_\gamma(y) \tilde{P}(y = y'(x)_{mle}|x) \rangle_{P(X)}}{\langle \tilde{P}(y = y'(x)_{mle}|x) \rangle_{P(X)}}$$

To test the effect of the likelihood term, MLE's $y'(x)_{mle}$ were calculated by a numerical optimisation procedure. Specifically, the Python SciPy [287, 204] implementations of Brent's algorithm was used (see §9.3 of [305]). An MLE tolerance parameter of $\epsilon = 1.0 \times 10^{-5}$ was selected to ensure the accuracy and numerical stability of the calculations. Numerical optimisation, rather than direct analytic calculation of the minimum at $y = \frac{x}{2}$ found by calculus, was used to make this problem closer in spirit to a more difficult optimisation problem that would require numerical evaluation.

To estimate the effect of the likelihood term on the computed MCS estimates, the joint probability distribution of the input and output spaces was integrated directly using the multivariate normal techniques in [138] using the SciPy wrapper to this function [204]. This was done to confirm the relative accuracy of the residual minimisation forward and Bayesian estimates. The results of these calculations were found to a tolerance of approximately 1×10^{-8} and presented to 6 significant figures in Table 6.1 under the column $P(y \geq \gamma)_{mle} \pm 1.0 \times 10^{-7}$.

6.5.1.3 Numerical results

The results of the MCS estimates by the traditional residual minimisation forward method and the Bayesian correction of the forward method are presented in Table 6.1. 1×10^6 simulations were used for all MCS estimates. These results show that the mean and variance of the traditional forward and Bayes corrected estimates are very similar, as would be expected given the unimodality of the energy surface and the discussion in Section 6.4.3. In fact, from the results in Table 6.1, differences only occur in the calculated variances on the tolerance of the MLE estimation, ϵ. This is a partial experimental confirmation of equations (6.71) using the MLE selecting proposal distribution in

equation (6.77). However, although both methods performed well for γ_0, neither method converged to a correct estimate for γ_1 and γ_2. This demonstrates that using MLE's to estimate Quantities of Interest far away from points of maximum likelihood (the mean of the joint probability in this problem) is not a good strategy. This failure is likely because of the poor performance of Importance Sampling predicted by equation (6.57) which essentially says that good performance requires the proposal distribution at each point to have a density profile that is close to the product of the true distribution and the random variable function of interest.

6.5.1.4 Improving the Quantity of Interest estimates

By realising that the traditional forward method is implicit Importance Sampling, the results of the Bayesian approach suggest an immediate improvement. Since the joint distribution is known, the joint distribution can be sampled from directly. The numerical results from the Bayesian forward estimate in Table 6.1 conform to expectations that the forward method performs better close to the mean. However, the 'Joint Distribution Sampling' (or 'JDS') method is vastly superior in this problem in all cases. The results of this analysis are presented in Table 6.1.

6.5.1.5 Discussion

It is well known that traditional forward estimates are not suitable for estimating the probability for an output to exceed some far from mean output [157]. By identifying the cause of this failure as a poor proposal distribution for sampling from the joint input-output space, improved estimation techniques can be applied to future problems. A simple example was used to demonstrate that these improvements are indeed possible. The values in Table 6.1 indicate that the traditional forward method fails to calculate an accurate QoI estimate for far from mean thresholds because the true value is outside of the estimated (surrogate) confidence intervals. This is because the actual variance in the estimate is very high. A combination of the Bayesian interpretation and Importance Sampling theory predict that the Dirac delta centred at MLE's will cause the proposal distribution to generate high variance estimates. The convergence of each method tested is also shown in Figure 6.2. The similarity

γ	$P(y \geq \gamma)_{mle}$ $\pm 1.0\times10^{-7}$	Method	MCS Estimate $= \overline{\psi_\gamma(y)}$ μ	σ
0	5.00000×10^{-1}	Forward	5.0140×10^{-1}	4.99998×10^{-1}
		Bayes	5.0140×10^{-1}	4.99996×10^{-1}
		JDS	4.9962×10^{-1}	5.00000×10^{-1}
1	1.58655×10^{-1}	Forward	2.3031×10^{-2}	1.50002×10^{-1}
		Bayes	2.3031×10^{-2}	1.50006×10^{-1}
		JDS	1.58853×10^{-1}	3.65539×10^{-1}
2	2.27501×10^{-2}	Forward	2.3000×10^{-5}	4.795778×10^{-3}
		Bayes	2.3000×10^{-5}	4.795776×10^{-3}
		JDS	2.2777×10^{-2}	1.491919×10^{-1}

Table 6.1: Numerical results comparing Bayesian and Forward estimates for the unimodal residual threshold problem. $P(y \geq \gamma)_{mle} \pm 1.0\times10^{-7}$ are the true values estimated as per [138] to a tolerance $\approx 1\times10^{-8}$ and presented to 6 significant figures. MCS estimates are shown for 1×10^{6} simulations. The 'Forward' method refers to the residual minimisation solution. The 'Bayes' method is the corrected estimate using the likelihood values. The 'JDS' method is sampling directly from the multivariate normal joint probability density described in Section 6.5.1.4. MLE's were calculated to a tolerance of 5 significant figures.

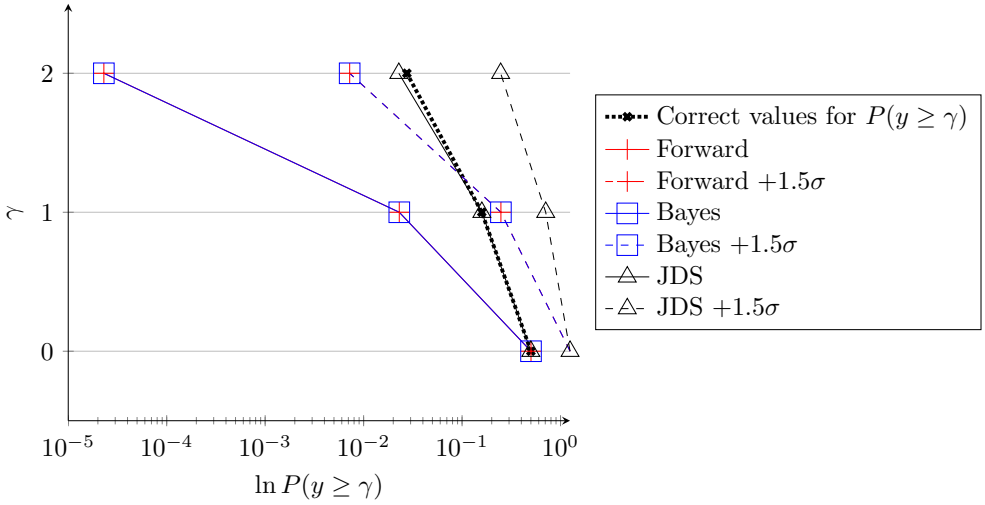


Figure 6.2: Unimodal threshold probability estimation problem results, comparing the performance of the residual minimisation forward method ('Forward'), the Bayesian likelihood corrected estimate ('Bayes') and joint distribution sampling ('JDS'). Solid lines indicate mean estimates. Dashed lines indicate $+1.5\sigma$ as a surrogate for the true confidence intervals. The thick black line indicates the correct values to be estimated. Note that the 'JDS' mean estimate is almost identical to the true estimate line.

of the Bayesian and traditional forward estimates is encouraging because it numerically confirms the developments in Section 6.4 for the unimodal case.

A bimodal problem is tested in Section 6.5.2 for which the traditional forward and Bayesian estimates should not be expected to be equal because of the effect of the likelihood terms in equation (6.81).

Sampling directly from the joint input-output distribution performed drastically better in all cases. For realistic problems, the full joint distribution is not known and is likely to be far more complicated than the simple multivariate Gaussian used for $P(X, Y)$ in this demonstration. However, what is true is that the drastically improved performance suggests that avoiding MLE's on more complicated problems may yield better performance. By applying more advanced techniques, such as the cross-entropy method [319] or Hamiltonian Monte Carlo [55], to sample from the joint distribution without reliance on MLE's it may be possible to improve performance over traditional methods on more difficult problems.

6.5.2 Bimodal residual function - Gaussian Mixture Model

This numerical experiment introduces a bimodal residual function in the form of a Gaussian Mixture Model [305]. A single threshold quantity is to be estimated. $P(X, Y)$ is formed by the sum of two Gaussians, $\mathcal{N}_A$ and $\mathcal{N}_B$, as shown in Figure 6.3. The global residual minimum is close to the mean of A. However, the threshold quantity ψ_γ to be estimated passes through the mean of $\mathcal{N}_B$. This is also demonstrated in Figure 6.4. $\mathcal{N}_A$ and $\mathcal{N}_B$ are far enough apart that ψ_γ should essentially only depend on the mass distribution due to $\mathcal{N}_B$, the influence of A simply acting like a constant factor on the probability of ψ_γ. Sampling ψ_γ using MLE's as in the traditional forward method can be expected to be produce poor results as the sampling locations will be near the mean of $\mathcal{N}_A$ and far from γ. After demonstrating that this is the case, an improved estimate based on samples near the mean of $\mathcal{N}_B$ is presented. This numerical experiment demonstrates the inability of forward methods based on residual minimisation to properly consider the joint space. Such a failure could have extreme consequences in, for example, reliability engineering or when interpreting experimental results. This example demonstrates that by taking a Bayesian view and considering the residual conditional distribution, Importance Sampling can be leveraged to improve the convergence of QoI estimates. Further, it is shown that sampling directly from the joint input-output space can be used to calculate accurate QoI estimates without reference

to MLE's.

6.5.2.1 Problem specification

With reference to equation (6.94), let $P(X,Y)$ be the mixture of two Gaussians, $\mathcal{N}_A$ and $\mathcal{N}_B$ such that:

$$P(X,Y) = \frac{1}{2}\mathcal{N}_A + \frac{1}{2}\mathcal{N}_B = \frac{1}{2}\mathcal{N}\left(\mu_A, \Sigma_A\right) + \frac{1}{2}\mathcal{N}\left(\mu_B, \Sigma_B\right) \tag{6.101}$$

where:

$$\mu_A = \begin{bmatrix} 0 \\ -2 \end{bmatrix} \quad \Sigma_A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mu_B = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad \Sigma_B = \begin{bmatrix} 1.5^2 & 0 \\ 0 & 1 \end{bmatrix}$$

Note that there are no correlation terms in $\mathcal{N}_A$ or $\mathcal{N}_B$ so x and y are independent in each of the mixed distributions:

$$\mathcal{N}_A = P_A(X)P_A(Y) \qquad \mathcal{N}_B = P_B(X)P_B(Y) \tag{6.102}$$

Then, with reference to equation (6.96), the input distribution can be found by:

$$\begin{aligned} P(X) &= \sum_{y \in Y} P(X,Y) \\ P(X) &= \sum_{y \in Y} \frac{1}{2}\mathcal{N}_A + \sum_{y \in Y} \frac{1}{2}\mathcal{N}_B \\ P(X) &= \frac{1}{2}P_A(X) + \frac{1}{2}P_B(X) \end{aligned}$$

Finally, $P(X)$ is:

$$P(X) = \frac{1}{2}\mathcal{N}(0,1) + \frac{1}{2}\mathcal{N}(0,1.5^2) \tag{6.103}$$

The problem is to evaluate, given $P(X)$, the threshold indicator Quantity of Interest $\psi_\gamma(y)$ for $\gamma = 2$ on the output space:

$$\psi_\gamma(y) = \chi_{y \geq (\gamma=2)}(y) \tag{6.104}$$

To evaluate the performance of the traditional forward method, an equation

for $H(y|x)$, the residual to be minimised by an optimisation procedure, is required. The residual function was derived from $P(X,Y)$ by:

$$P(Y|X) = \frac{P(X,Y)}{P(X)}$$
$$P(Y|X) = \frac{\frac{1}{2}\mathcal{N}_A + \frac{1}{2}\mathcal{N}_A}{\frac{1}{2}P_A(X) + \frac{1}{2}P_B(X)}$$
$$P(y|x) = \frac{\frac{1}{2\pi}e^{-\frac{x^2}{2}-\frac{(y+2)^2}{2}} + \frac{1}{2\pi\times 1.5^2}e^{-\frac{x^2}{2\times 1.5^2}-\frac{(y-2)^2}{2}}}{\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} + \frac{1}{\sqrt{2\pi\times 1.5^2}}e^{-\frac{x^2}{2\times 1.5^2}}}$$

so that $P(y|x)$ is given by:

$$P(y|x) = \frac{\frac{1}{2\pi}e^{-\frac{x^2}{2}-\frac{(y+2)^2}{2}} + \frac{1}{4.5\pi}e^{-\frac{x^2}{4.5}-\frac{(y-2)^2}{2}}}{\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} + \frac{1}{\sqrt{4.5\pi}}e^{-\frac{x^2}{4.5}}} \tag{6.105}$$

then the residual function is found calculating:

$$H(y|x) = -\ln P(y|x) \tag{6.106}$$

6.5.2.2 Numerical results

To test the solutions calculated by sampling, first the known $P(X,Y)$ function was used to directly calculate $\langle\psi_\gamma(y)\rangle$. Intuitively, by inspection of Figure 6.3 (and because the problem was intentionally constructed in this way) the expected value of $\langle\psi_\gamma(y)\rangle \approx 0.25$ as half of the joint probability mass is contained in $\mathcal{N}_A$ (which is concentrated far from γ) and because γ lies on the mean of $\mathcal{N}_B$. To confirm this intuition, the total integral was calculated by the method for multivariate Gaussian integration described in [138] to a tolerance of approximately 1×10^{-8}. The following values were calculated:

$$\langle\psi(y)\rangle_{\mathcal{N}_A(X,Y))} = 3.16712579\times10^{-5} \pm 1.45862172\times10^{-8}$$
$$\langle\psi(y)\rangle_{\mathcal{N}_B(X,Y))} = 5.00000000\times10^{-1} \pm 1.46002488\times10^{-8}$$
$$\langle\psi(y)\rangle_{P(X,Y))} = 2.50015836\times10^{-1} \pm 1.47604643\times10^{-8}$$

For the purposes of this experiment, the solution of the bimodal threshold

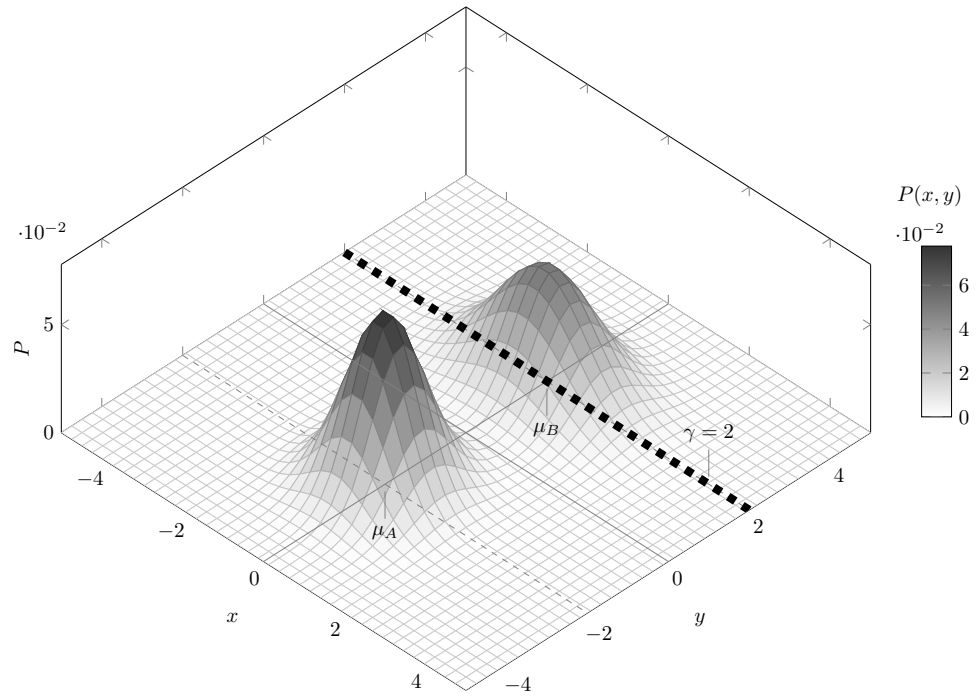


Figure 6.3: Bimodal threshold probability estimation problem overview: find $P(y \geq 2) =?$. μ_A and μ_B are the means of the Gaussians $\mathcal{N}_A$ and $\mathcal{N}_B$ which are evenly mixed to form $P(X, Y)$.

evaluation problem can be taken to be approximately as follows:

$$\langle \psi(y) \rangle_{P(X,Y))} \approx 2.50016 \times 10^{-1} \tag{6.107}$$

Next, the traditional forward estimate using global residual minimisation was calculated. The inputs were sampled from the input probability given in equation (6.103). Sampling from the mixture was achieved by first sampling, u from a uniform distribution on $[0, 1]$, selecting $\mathcal{N}_A$ if $u \leq 0.5$ (otherwise selecting $\mathcal{N}_B$) and then sampling from the selected normal distribution by the standard inverse transform method [224]. The residual was calculated using equations (6.105) and (6.106). Forward MLE's were evaluated using the Python SciPy implementation of the Basin Hopping algorithm (see [287, 204], §9.3 in [391]). The results of these calculations are presented in Table 6.2 in the row with method column labelled 'Forward'. The Bayesian estimate of the forward method performance, factoring in the likelihood term, as in equation (6.81), is also presented in in Table 6.2 in the row with method column labelled 'Bayes'.

Note that although the Bayes method estimates a smaller variance than the implicit Bayes in the forward method, the variance of the normalisation constant $\langle \tilde{w}(a) \rangle_{P(X)}$ was found to be approximately 3.8533×10^{-2}. This value is very small and will introduce numerical instability in the variance calculation. In either case, both the Bayes and MLE forward method predict $\psi_\gamma(y)$ probabilities very far from the true value.

6.5.2.3 Improving the traditional forward method

From the definition of the problem and by inspection of Figure 6.3, it is clear that a better choice of proposal distribution would be to sample from $\mathcal{N}_B$ directly, without reference to intermediate MLE's. However, a forward Uncertainty Quantification problem is typically specified by its input distribution and the residual function $H(y|x)$. It is not the case that a problem is constructed based on a known answer, as is done when building numerical experiments to empirically test some feature from theory. Then, without resorting to variational methods such as cross entropy [319] which would be required for high dimensional spaces, how could a useful proposal distribution that does not rely on MLE's be found for this simple two dimensional problem given just the input and residual function? Figure 6.4 shows a plot of the residual surface $H(y|x = 0)$ and it's associated probability distribution $P(y|x = 0)$. The graph suggests sampling close to $\gamma = 2$ as it is both close to the quantity of interest and has a reasonably low energy surface (and high likelihood probabilities) in this area. The conditional distribution and residual are calculated using equation (6.105). Then, for Figure 6.4 at $x = 0$:

$$P(y|x=0) = \frac{\frac{1}{2\pi} e^{-\frac{(y+2)^2}{2}} + \frac{1}{4.5\pi} e^{-\frac{(y-2)^2}{2}}}{\frac{1}{\sqrt{2\pi}} + \frac{1}{\sqrt{4.5\pi}}} \tag{6.108}$$

Following from the above discussion, the proposal probability, $Q_{BI}(y|x = 0)$, termed the 'Bayes Improved' distribution, on the joint space is selected to be:

$$Q_{BI}(x, y) = \mathcal{N}_B = \mathcal{N}(\mu_B, \Sigma_B) \tag{6.109}$$

The results of MCS using equation (6.109) as a proposal distribution are presented in Table 6.2 in the row labelled by the method 'BI'.

Finally, the joint space distribution is sampled from directly. The results of this 'Joint Distribution Sampling' are shown in Table 6.2 in the row labelled by the method 'JDS'.

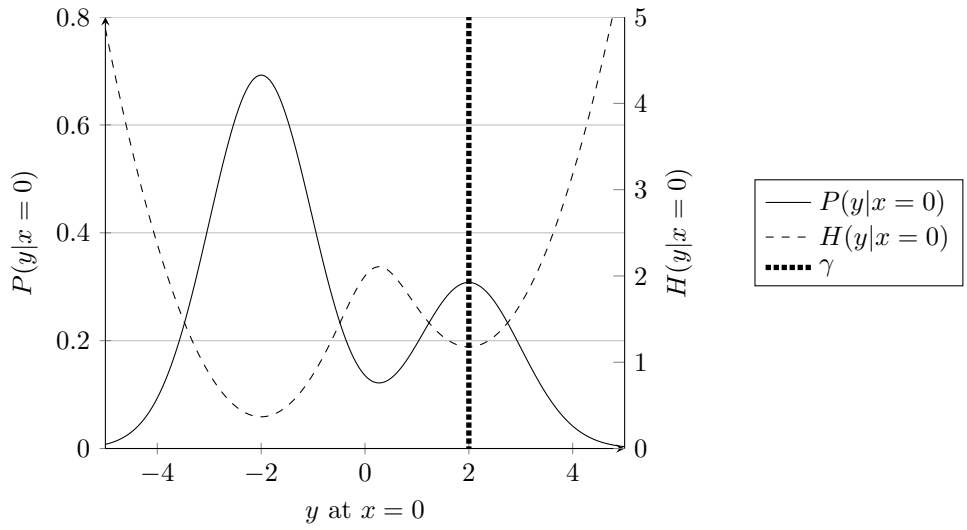


Figure 6.4: Residual energy $H(y|x=0)$ and probability density $P(y|x=0)$ for the bimodal estimation problem $P(y \geq \gamma = 2) =?$. Note that the global energy minimum is at μ_A, away from $\gamma = 2$. The concentration of density at γ due to $\mathcal{N}_B$ suggests setting $Q(x, y) = \mathcal{N}_B$.

6.5.2.4 Discussion

This example demonstrated several issues. First, the traditional forward method fails to account for probability mass near a threshold that could have been detected by considering the residual function. The forward method therefore underestimates the probability of threshold indicator. Such an underestimate could be disastrous in a reliability problem if the threshold was used to describe, for example, the collapse state of a dam or the change for some dangerous run-away reaction to occur. Overestimating reliability in this way is avoidable by considering the structure of the residual function as well as the structure of the input probability space. This example also demonstrated that the Bayesian interpretation of the forward method is able to detect potential failures of the traditional forward method by considering the expected value of the normalisation constant. This suggests that if a forward method based on MLE's is used, the Bayesian normalisation constant may be a useful surrogate to test the accuracy of the forward method. MLE's may still be useful when

γ	$P(y \geq \gamma)_{mle}$ $\pm 1.0\times10^{-7}$	Method	MCS Estimate = $\overline{\psi_\gamma(y)}$ μ	σ
2	2.50016×10^{-1}	Forward	2.3100×10^{-4}	1.5197×10^{-2}
		Bayes	1.5209×10^{-8}	6.2825×10^{-4}
		BI	2.1820×10^{-1}	3.4981×10^{-1}
		JDS	2.4951×10^{-1}	4.3273×10^{-1}

Table 6.2: Numerical results comparing Bayesian and Forward estimates for the bimodal residual threshold problem. $P(y \geq \gamma)_{mle} \pm 1.0\times10^{-7}$ are the true values after mixing $\mathcal{N}_A$ and $\mathcal{N}_B$ values calculated as per [138] to a tolerance $\approx 1\times10^{-8}$, presented to 6 significant figures. MCS estimates are shown for 1×10^6 simulations. The 'Forward' method refers to the residual minimisation solution. The 'Bayes' method is the corrected estimate using the likelihood values. The 'BI' method is Importance Sampling over the joint input-output space from Section 6.5.2.3. The 'JDS' samples directly from the joint space distribution. Note that the low variance of the 'Bayes' estimate is likely due to numerical instability, see Section 6.5.2.2.

the structure of the residual function is much more complicated than the problem given here. An example of this approach would be using a combination of MLE's to find low residual parts of the joint space followed by Hamiltonian Monte Carlo [55] for Markov chain sampling of the joint space.

Finally, this problem demonstrates that by taking a Bayesian view and considering the structure of the joint input-output space, both Importance Sampling and sampling from the joint input-output space can be leveraged to improve the convergence of QoI estimates without reference to MLE's at all. By sampling near the threshold indicator function even with a suboptimal proposal distribution, a much more reasonable estimate of the actual threshold probability was found. This avoids the reliability overestimation problem. This numerical experiment identified a situation that causes the traditional forward Uncertainty Quantification method to fail (due to bimodality of the output space distribution) that can be rectified by considering the joint input-output space directly. This is likely to be useful for future work in Uncertainty Quantification.

6.5.3 Multimodal problem - threshold estimation with complicated residual function

The final numerical experiment presented asses the effect of a multimodal residual surface on the traditional forward method based on MLE's. The so-

called 'Bird function' is used to define $H(y|x)$. As well as multimodal, this function is also continuous, differentiable and non-separable [200]. As in the previous numerical experiments in this Section, a threshold indicator function is used as a QoI. This problem again demonstrates that the traditional forward Uncertainty Quantification method based on MLE's may be suboptimal as these MLE's effectively constitute Importance Sampling with a poor choice of proposal distribution.

Additionally, the effect of the variance of the input distribution on the joint energy surface, $H(x, y)$, is demonstrated. It is shown that reducing the input distribution variance (for Gaussian distributions) smooths out the global energy surface which, in the case of the problem tested, may render MLE's on the featureless joint energy surface even less useful for estimating a given QoI. Further, as the structure of $H(x, y)$ is then far simpler than $H(y|x)$, this suggests that Markov Chain Monte Carlo can be used to efficiently sample from the joint space without resorting to computationally expensive MLE's. Markov Chain Monte Carlo on the joint input-output space is shown to be effective for calculating the QoI estimate in this problem.

6.5.3.1 Problem Specification

Let $H(y|x)$ be given by the so-called 'Bird function' [200]:

$$H(y|x) = (x - y)^2 + e^{[1-\sin(x)]^2}\cos(y) + e^{[1-\cos(y)]^2}\sin(x) \tag{6.110}$$

restricted to $x \in [-2\pi, 2\pi] \subset \mathbb{R}$ and $y \in [-2\pi, 2\pi] \subset \mathbb{R}$. A plot of this highly multimodal residual surface is shown in Figure 6.5a. The Bird function has a global minimum of approximately -110. As such, for numerical stability, a constant value of 110 has been added on to all evaluations of the residual made when solving this problem.

Two input distributions, $P_A(X)$ and $P_B(X)$ are specified:

$$P_A(X) = \mathcal{N}_A(0, 0.5^2) \tag{6.111}$$

$$P_B(X) = \mathcal{N}_B(0, 0.3^2) \tag{6.112}$$

The problem is to evaluate, given $P(X)$, the threshold indicator Quantity of

Interest $\psi_\gamma(y)$ for $\gamma = 0$ on the output space:

$$\psi_\gamma(y) = \chi_{y \geq (\gamma=0)}(y) \tag{6.113}$$

6.5.3.2 Numerical results

The solution of the specified problem are given by integrals of $P(X,Y) = P(Y|X)P(X)$ for each input distribution $P_A(X)$ and $P_B(X)$ [224]:

$$\psi_\gamma(y) = \int_{-2\pi}^{2\pi} \int_{0}^{2\pi} P(X,Y) dx dy = \frac{\int_{-2\pi}^{2\pi} \int_{0}^{2\pi} e^{-H(y|x)} P(X) dy dx}{\int_{-2\pi}^{2\pi} \int_{-2\pi}^{2\pi} e^{-H(y|x)} P(X) dy dx} \tag{6.114}$$

Unfortunately, it is very difficult to evaluate equation (6.114) by some direct form of quadrature, as was used the other numerical experiments in Section 6.5.1 and 6.5.2. As such, no independent solutions of the threshold probability problem are supplied in this case.

The traditional forward estimate using global residual minimisation was calculated for each of the input distributions in equations (6.111) and (6.112). The residual was calculated using equation (6.110). Forward MLE's were evaluated using the Python SciPy implementation of the Basin Hopping algorithm (see [287, 204], §9.3 in [391]). The results of these calculations are presented in Table 6.3 in the row with method column labelled 'Forward'. For this problem, the Bayesian calculation of the forward method likelihoods is very numerically unstable and has been excluded form Table 6.3. This indicates only that the delta function makes numerical estimates difficult, not any problem with the Bayesian interpretation of Uncertainty Quantification detailed in this Chapter.

6.5.3.3 Effect of input variance on the joint energy surface

The effect of the changing input variance between $P_A(X)$ and $P_B(X)$ on the global energy function is shown in Figures 6.5b and 6.5c respectively. These Figures indicate that decreasing the variance of a Gaussian input distribution acts to smooth out the global energy surface. Where $P(X)$ has high probability mass, the joint energy surface is lowered and flattened out. This means that the global energy surface, unlike the conditional energy surface,

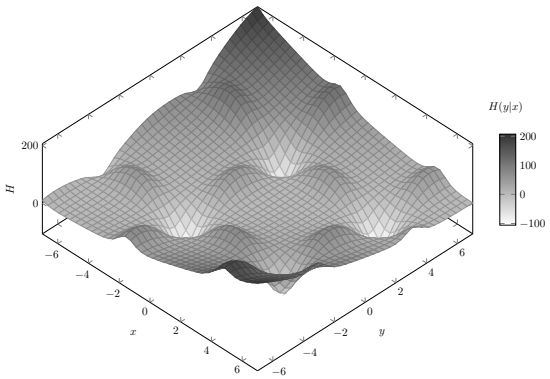
is not too multimodal and suggests that Markov Chain Monte Carlo techniques may be used to sample from the joint space effectively [55]. To this end, the performance of a simple Metropolis Hastings sampler was tested on estimates of $\psi_\gamma(y)$. The following Metropolis-Hastings proposal distribution was used:

$$Q(x, y) = \mathcal{N}\left(\mu, \begin{bmatrix} 0.1^2 & 0 \\ 0 & 0.1^2 \end{bmatrix}\right) \tag{6.115}$$

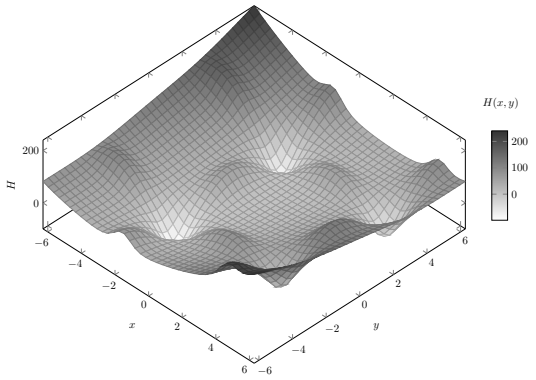
The results of the Markov Chain Monte Carlo sampling analysis are presented in Table 6.3 in the rows labelled by the method 'MCMC'. Note that for Markov Chain Sampling, samples outside of the analysis bounds were rejected and not incorporated into the QoI probability calculations. The Markov Chain Monte Carlo Sampler does not need to evaluate any MLE's and as such each sample can be generated very efficiently when compared to the traditional forward method. An acceptance ratio of approximately 0.5 was obtained during the analysis.

$P(X)$	Method	MCS Estimate = $\overline{\psi_\gamma(y)}$	
		μ	σ
$P_A(X) = \mathcal{N}(0, 0.5^2)$	Forward	6.4477×10^{-1}	4.7859×10^{-2}
	MCMC	6.7517×10^{-1}	4.6831×10^{-1}
$P_B(X) = \mathcal{N}(0, 0.3^2)$	Forward	6.3442×10^{-1}	4.8159×10^{-1}
	MCMC	6.8655×10^{-1}	$4.6390e\times10^{-1}$

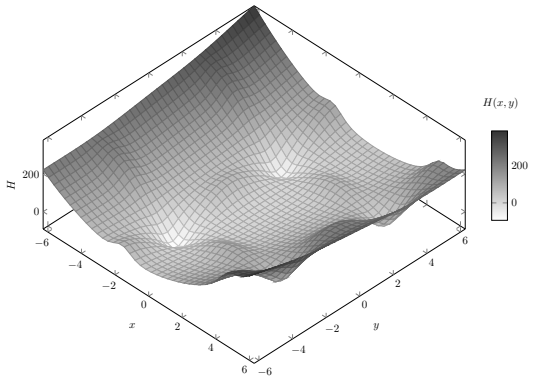
Table 6.3: Numerical results comparing Forward and Metropolis Hastings estimates for multimodal residual threshold problem. MCS estimates are shown for 1×10^6 simulations. The 'Forward' method refers to the residual minimisation solution. The 'MCMC' method refers to Metropolis Hastings sampling over the joint input-output space.



(a) $H(y|x) = (x - y)^2 + e^{[1-\sin(x)]^2}\cos(y) + e^{[1-\cos(y)]^2}\sin(x)$



(b) $H(x, y) = (H(y|x) - \ln P_A(X))$ for $P_A(X) = \mathcal{N}(0, 0.5^2)$



(c) $H(x, y) = (H(y|x) - \ln P_B(X))$ for $P_B(X) = \mathcal{N}(0, 0.3^2)$

Figure 6.5: Multimodal problem energy surfaces. Figure 6.5a shows the residual energy surface from equation (6.110). Figures 6.5b and 6.5c show the joint energy surface $H(x, y)$ with input distributions $P_A(X)$ and $P_B(X)$ from equations (6.111) and (6.112) respectively.

6.5.3.4 Discussion

For this problem, the traditional forward method performed reasonably well because the quantity to be estimated was near to the Quantity of Interest threshold boundary. However, the computation of each of the required MLE's is computationally expensive. Markov Chain Monte Carlo was shown, for this problem, to be an effective way to sample directly from the joint input-output space without any MLE evaluations. The effectiveness of the Markov Chain sampler is likely because of the flattening of the joint global surface demonstrated in Figure 6.5. More advanced Markov Chain methods such as Hamiltonian Monte Carlo could also be applied to potentially achieve better QoI estimates still while still retaining the speed advantage over the forward method [55]. This is particularly likely to be the case for rare event estimation which requires techniques such as Subset Simulation [15, 16, 157]. This numerical example further demonstrates that MLE's are not required and that the traditional forward method may not be the most computationally efficient procedure for Uncertainty Quantification.

6.6 Conclusions

This Chapter explored the links between traditional Uncertainty Quantification methods based on the pushforward measure, the Bayesian interpretation of Uncertainty Quantification and solving equations by residual minimisation. Residual minimisation was shown to be a form of Maximum Likelihood estimation. Further, it was demonstrated that the traditional Uncertainty Quantification methods based on residual minimisation of the solutions of equations are in fact an implicit form of Importance Sampling. This result is significant because it identifies the cause of the failure of traditional Uncertainty Quantification techniques for multimodal problems and for far from mean Quantity of Interest estimates. By including the likelihood to have sampled a minimum residual solution in the first place, the correct variance of an estimate of a Quantity of Interest can be recovered. Further, the cause of the failure of traditional Uncertainty Quantification methods for certain problems was identified as a poor choice of Importance Sampling proposal distribution. This suggests that alternative sampling distributions can be selected to improve QoI estimate convergence. Numerical experiments were also conducted, demonstrating the

consequences of this implicit Importance Sampling for probabilistic problems. These examples demonstrated the theoretical findings empirically. Moreover, these numerical experiments demonstrated the potential for the traditional forward to Uncertainty Quantification approach to overestimate reliability of systems when the residual energy function is multimodal. The developments presented will help find improved Uncertainty Quantification methods by improving the understanding of the Bayesian interpretation of solving equations in future applications and research.

Chapter 6 Appendix: Proof of Theorems 6.6.1 and 6.6.2 - Forward estimate proposal distribution is valid

Theorem 6.6.1. Given QoI function $\psi : Y \to \Psi$, the proposal distribution:

$$\tilde{Q}(x, y) = \delta \left(y - \underset{y \in Y}{\operatorname{argmax}} \, P(y|x) \right) P(x) \tag{6.116}$$

and the joint input-output distribution:

$$P(x, y) = P(y|x)P(x) \tag{6.117}$$

the following is true:

$$Z_{Q(X,Y)} = \sum_{x \in X} \sum_{y \in Y} \tilde{Q}(x, y) > 0 \tag{6.118}$$

Proof. First, define:

$$y'(x) := \underset{y \in Y}{\operatorname{argmax}} \, P(y|x) \tag{6.119}$$

$$\begin{aligned} Z_{Q(X,Y)} &= \sum_{x \in X} \sum_{y \in Y} \tilde{Q}(x, y) \\ Z_{Q(X,Y)} &= \sum_{x \in X} \sum_{y \in Y} \delta \left(y - y'(x) \right) \right) P(x) \\ Z_{Q(X,Y)} &= \sum_{x \in X} P(x) \end{aligned}$$

but $P(X)$ is a probability distribution so $\sum_{x \in X} P(x) = 1$ therefore:

$$Z_{Q(X,Y)} = \sum_{x \in X} P(x) = 1 > 0 \tag{6.120}$$

□

Theorem 6.6.2. Given QoI function $\psi : Y \to \Psi$, the proposal distribution:

$$\tilde{Q}(x, y) = \delta \left(y - \underset{y \in Y}{\operatorname{argmax}} \, P(y|x) \right) P(x) \tag{6.121}$$

and the joint input-output distribution:

$$P(x, y) = P(y|x)P(x) \tag{6.122}$$

the following is true:

$$N(x, y) := \psi(y)P(x, y) \neq 0 \Rightarrow Q(x, y) > 0 \tag{6.123}$$

Proof. First, note that as $Q(x, y)$ is a probability measure with minimum value 0, $Q(x, y) \leq 0 \Leftrightarrow Q(x, y) = 0$.

Also note that from Theorem 6.6.1, $Q(x, y)$ is normalised so $\tilde{Q}(x, y) = Q(x, y)$.

For the main part of the proof, assume that $N(x, y) \neq 0$. Then the proof is satisfied if $Q(x, y) > 0$ for all x, y. If $N(x, y)$, then $\psi(y)P(y|x)P(x) \neq 0$ implies that $\psi(y) \neq 0$, $P(y|x) \neq 0$ and $P(x) \neq 0$:

$$N(x, y) \neq 0 \iff \psi(y)P(y|x)P(x) \neq 0 \tag{6.124}$$

$$\psi(y)P(y|x)P(x) \neq 0 \iff \psi(y) \neq 0, P(y|x) \neq 0, P(x) \neq 0 \tag{6.125}$$

Then the proof is satisfied if each of:

$$\psi(y) \neq 0 \overset{?}{\Longrightarrow} Q(x, y) > 0 \qquad \forall x \in X, \forall y \in Y \tag{6.126}$$

$$P(y|x) \neq 0 \overset{?}{\Longrightarrow} Q(x, y) > 0 \qquad \forall x \in X, \forall y \in Y \tag{6.127}$$

$$P(x) \neq 0 \overset{?}{\Longrightarrow} Q(x, y) > 0 \qquad \forall x \in X, \forall y \in Y \tag{6.128}$$

is satisfied.

For equation (6.126), both $\psi(y) = 0$ and $\psi(y) \neq 0$ satisfy $Q(x, y) > 0$ so:

$$[\psi(y) \neq 0] \ \& \ [\psi(y) = 0] \implies Q(x, y) > 0$$

is true by the definition of material entailment.

It will be useful to first consider the conditions under which:

$$\delta \left(y - \underset{y \in Y}{\operatorname{argmax}} \ P(y|x) \right) \overset{?}{>} 0 \tag{6.129}$$

For convenience, define:

$$y'(x) := \underset{y \in Y}{\operatorname{argmax}} \ P(y|x) \tag{6.130}$$

Let $x' \in X$ be some arbitrary element of X. Using the definition of the argmin operator (Definition 6.2.8), $y'(x') \in Y$ so $y'(x) \in Y \ \forall x \in X$.

Also consider that from the definition of the delta function:

$$\delta(y - y(x)) > 0 \ \forall x \in X, y \in Y \Rightarrow \exists y \in Y : \delta(y - y'(x)) = 1 \ \forall x \in X \tag{6.131}$$

Then it follows that if $\delta(y - y'(x))$ is defined $\forall y \in Y$ and $y'(x) \in Y \ \forall x \in X$ the following statements are true:

$$\begin{gathered}
\delta \left(y - y'(x) \right) = 0 \ \forall x \in X, \forall y \in Y \implies y \neq y'(x) \ \forall x \in X, \forall y \in Y \\
\neg \left[\delta \left(y - y'(x) \right) = 0 \ \forall x \in X, \forall y \in Y \right] \implies \neg \left[y \neq y'(x) \ \forall x \in X, \forall y \in Y \right] \\
\exists y \in Y : \delta \left(y - y'(x) \right) = 1 \ \forall x \in X \implies \exists y \in Y : y = y'(x) \ \forall x \in X
\end{gathered}$$

If $\delta(y - y'(x))$ defined over the entire space $y \in Y$, then as $Q(y|x) = \delta(y - y'(x))$ and $\sum_{y \in Y} Q(y|x) = 1$ by the definition of a probability measure, then:

$$\begin{gathered}
\sum_{y \in Y} Q(y|x) = 1 \\
\sum_{y \in Y} \delta(y - y'(x)) = 1 \implies y = y'(x)
\end{gathered}$$

This implies the following is true, if $y'(x) \in Y \ \forall x \in X$:

$$\exists y \in Y : \delta(y - y'(x)) = 1 \ \forall x \in X \tag{6.132}$$

Next, consider the condition on $P(x)$ required by equation (6.128). First, if $Q(x, y) = \delta(y - y'(x))P(x) > 0$ and $P(x) \neq 0$ implies:

$$\begin{aligned} P(x) \neq 0 \ \forall x \in X &\overset{?}{\Longrightarrow} Q(x, y) > 0 \ \forall x \in X, \ \forall y \in Y \\ &\Longrightarrow \delta(y - y'(x))P(x) > 0 \ \forall x \in X, \ \forall y \in Y \\ &\Longrightarrow \delta(y - y'(x)) > 0 \ \forall x \in X, \ \forall y \in Y \\ &\Longrightarrow \exists y \in Y : \delta(y - y'(x)) = 1 \ \forall x \in X \end{aligned}$$

From equation (6.132), $\exists y \in Y : \delta(y - y'(x)) = 1 \ \forall x \in X$ is always true for all X so that:

$$P(x) \neq 0 \Longrightarrow \exists y \in Y : \delta(y - y'(x)) = 1 \ \forall x \in X$$

and equation (6.128) is true.

Finally, it remains to confirm equation (6.127). First consider that:

$$\begin{aligned} Q(x, y) \neq 0 &\Longrightarrow \delta(y - y'(x))P(x) \neq 0 \qquad \forall x \in X, \forall y \in Y \\ \delta(y - y'(x))P(x) \neq 0 &\Longrightarrow \delta(y - y'(x)) \neq 0 \\ \delta(y - y'(x))P(x) \neq 0 &\Longrightarrow P(x) \neq 0 \end{aligned}$$

Then equation (6.127) can be split into two parts:

$$P(y|x) \neq 0 \overset{?}{\Longrightarrow} P(x) \neq 0 \qquad \forall x \in X, \forall y \in Y \tag{6.133}$$

$$P(y|x) \neq 0 \overset{?}{\Longrightarrow} \delta(y - y'(x)) > 0 \qquad \forall x \in X, \forall y \in Y \tag{6.134}$$

First, consider that equation (6.134) can expanded:

$$\begin{aligned} P(y|x) \neq 0 \ \forall x \in X, \forall y \in Y &\overset{?}{\Longrightarrow} \delta(y - y'(x)) > 0 \ \forall x \in X, \forall y \in Y \\ P(y|x) \neq 0 &\overset{?}{\Longrightarrow} \exists y \in Y : \delta(y - y'(x)) = 1 \ \forall x \in X \end{aligned}$$

which is always true by equation (6.132) so:

$$P(y|x) \neq 0 \Longrightarrow Q(x,y) > 0 \ \forall x \in X, \forall y \in Y$$

Finally, consider equation (6.133). Using the joint and marginal probabilities:

$$\begin{aligned} P(x) &= \sum_{y \in Y} P(x,y) \\ P(x) &= \sum_{y \in Y} P(y|x)P(x) \\ P(x) \neq 0 &\Longleftrightarrow \sum_{y \in Y} P(y|x)P(x) \neq 0 \\ \sum_{y \in Y} P(y|x)P(x) \neq 0 &\Longleftrightarrow \exists y \in Y : P(y|x) \neq 0 \end{aligned}$$

Then, equation (6.133) is true:

$$P(y|x) \neq 0 \Longrightarrow P(x) \neq 0 \qquad \forall x \in X, \forall y \in Y$$

As each of the equations (6.126), (6.127) and (6.128) are true, by equation (6.125) the required statement, equation (6.123), is true. □

Chapter 7

Adaptive Element Free Galerkin via Bayesian Probability and Artificial Neural Networks

Contents

Figures

Chapter 7 Overview

Key developments in Chapter 7 include:

- Section 7.2 provides background material on the Element Free Galerkin Method for solving PDE problems.
- Section 7.2.4 discusses parametric representations of basis functions for PDE solutions Artificial Neural Networks, which is used for the main result of this Chapter.

- Section 7.3 presents background theory in Bayesian Inference and optimisation that is necessary to understand the main developments in this Chapter.
- Section 7.4 presents an original contribution of this thesis: an adaptive basis Element Free Galerkin method based on a probabilistic interpretation of the Galerkin method. ANNs are used to parametrically represent basis functions and an iterative algorithm, analogous to Expectation-Maximisation, is used to derive a two-stage optimisation method for the updating basis functions.
- Section 7.5 presents a simple numerical example to demonstrate the new algorithm proposed in this Chapter.

7.1 Introduction

This Chapter demonstrates that Artificial Neural Networks (ANNs) and Automatic Differentiation can be used to augment weak-form Element Free Galerkin solutions of Partial Differential Equations in a self-improving, adaptive manner. In particular, it is demonstrated that by representing the solution of a PDE using the Bayesian form of parameter regression, an objective function for the quality of a basis set can be given and, therefore, optimised. In Bayesian terms, this objective function is related to the model evidence (or likelihood) for the estimated PDE solution given the current basis function set. The objective function for the basis optimisation is derived in terms of the Expectation-Maximisation algorithm. The formulation presented also uses sparse-coding to enforce independence of the learnt basis functions. The basis functions are represented parametrically by an Artificial Neural Network. The iterative refinement procedure derived within allows for the basis function parameters to be trained directly by standard backpropagation. Boundary constraints are the primary cause of difficulties when finding PDE solutions by optimisation methods. Constrained optimisation is achieved in this context using an Expectation-Maximisation method by splitting the optimisation into two parts. First, the Galerkin PDE formulation is solved using the current basis functions. Second, optimisation of the basis functions is achieved by backpropagation using the current solution estimate. This technique is demonstrated on a one dimensional Poisson problem. The numerical results indicate

that the proposed algorithm, referred to sparse-coding Bayesian Element Free Galerkin, is able to iteratively reduce the error in the estimated PDE solution. Although only linear problems are considered in this Chapter, the results presented should facilitate the development of adaptive nonlinear PDE solvers over higher dimensional domains in the future. Further, by uncovering the Bayesian Inference interpretation of Galerkin methods in detail, directions for future improvements to automated PDE solvers are suggested.

This Chapter deviates slightly from the Uncertainty Quantification focus of the rest of this thesis in that the PDE solution methodology presented here is intended to solve deterministic problems. However, consider the supervised learning approach to probabilistic PDE problem response surfaces presented in Chapter 5. In Chapter 5, PDE problems were solved by the Finite Element Method and then an ANN map from the space of inputs to the space of these FEM solutions for a given problem was learnt. If the FEM solver could be replaced by an ANN based method, then it would potentially be possible to learn an integrated solver and response surface model by using a single large ANN to capture essentially all dependencies within a problem. Such an approach would be advantageous in that the burden of feature and solver design would be minimised as the PDE solutions and surrogate model could be learnt simultaneously. Although this ambitious goal is not fully realised for now, this Chapter addresses a critical sub-problem on that path by demonstrating that deterministic PDE problems can be solved by ANN auto-adaptive methods.

Probabilistic numerics uses Bayesian Inference as a lens through which to interpret the structure of the numerical solutions of PDE's and other problems such as integration and optimisation [175]. The unknown solution of some numerical method is treated as a hidden variable. Bayesian Inference can then be applied to understand procedures by which the values of such hidden variables can be estimated. The Bayesian framework allows for the quality of different solution procedures to be compared via the model evidence equations [275]. By introducing the probabilistic framework, it is possible to understand the algorithms for the solution of boundary value PDEs (constrained optimisation [198]) in terms of Maximum a Posteriori (MAP) estimates of the hidden solution function. Constrained optimisation is a MAP, rather than Maximum Likelihood Estimation (MLE), problem in the sense that the boundary

constraints can be represented in terms of prior beliefs about the unknown solution function, as discussed in Section 7.4 (although fine distinction between the two terms is not especially critical). Introducing constraints makes optimisation more challenging [41]. Further, when dealing with PDEs, it is necessary to compute function derivatives. This further restricts the range of suitable optimisation methods.

The Element Free Galerkin (EFG) method provides a method for finding PDE boundary value problem solutions [33] and is described in more detail in Section 7.2. The standard Galerkin Finite Element Method is to find the projection of a PDE onto a set of basis functions (termed the weak-form of a solution) and then derive an optimisation problem from the vanishing of the first functional derivative of the weak-form solution. The EFG method extends this approach by considering more arbitrary basis functions. A Lagrange Multiplier approach can be used to introduce boundary constraints [19]. The selection of basis functions is usually based on either locally supported functions (as in the standard Finite Element Method [78] or orthogonal series expansions (as in Spectral Element Methods [208]). These basis functions are typically selected based on two criteria. First, derivatives of the basis functions must be calculated and so simple basis functions are preferred. Second, integrals using the basis functions must be carried out. Basis functions with known integral forms, or those that are amenable to a quadrature procedure, are leveraged to simplify these integrals. When considering adaptive basis functions, the standard approach is to add more basis functions. PDE solutions are improved either by adding additional locally supported functions (h-refinement) or by increasing the order of the series expansion representing the basis (p-refinement) [78, 20]. These methods can be combined to varying degrees and are both, in a sense, based on the convergence properties of series expansions.

This Chapter introduces the use of ANN parametric basis function models. The adjustable basis function parameters allow for the shape of the basis functions to be altered, given some measure of the quality of the current basis functions, to best suit the PDE problem. Modern ANN implementations utilise Automatic Differentiation to simplify the calculation of the various derivatives required to fit ANNs. Automatic Differentiation can thus be used to find the basis function derivatives required to represent Partial Differential

Equations. Integration of this arbitrary, adjustable, basis function model is more challenging. A Monte Carlo Integration procedure is adopted for this purpose in this Chapter. Another ANN approach to solving constrained optimisation PDE problems is CPROP, discussed in [120, 321]. The CPROP method uses a more complicated form a ANN training than the direct back-propagation method. Further, the method presented in this Chapter can easily be used with deep ANNs, providing a very flexible representation of PDE solution basis functions. Given an adjustable ANN model, along with the EFG method to solve a boundary value PDE given a set of basis functions, the final outstanding issue is the derivation of some optimisation objective for improving the basis functions. Utilising Bayesian probability, such a basis training objective function can be derived.

Bayesian Inference provides a way to develop a basis function training objective in terms of model likelihoods. In particular, the Expectation-Maximisation (EM) algorithm [92, 259, 278] is utilised to derive an optimisation objective. The EM algorithm provides an iterative update procedure to calculate MLE or MAP estimates of a probabilistic model with latent variables. From a probabilistic perspective, fitting basis function regression models is a form Kernelised Bayesian Regression [46] and is thus related to Factor Analysis [275], Gaussian Processes [308] and Information Theory (in particular the Kullback-Liebler Divergence) [335]. The EM adaptive basis algorithm is detailed in Section 7.4. To summarise the method, consider a given likelihood function that represents the quality of a PDE solution (specified as a part of the PDE problem description). A latent variable model of the solution function can be derived in terms of Bayesian Regression. Both the basis functions and the regression weights for the basis functions are taken to be latent variables. EM allows for the regression weights and the basis function parameters to be updated iteratively, using the current parameters to estimate better parameter values. Introducing an EFG formulation further simplifies the optimisation problem by providing a clear way to introduce boundary constraints. Finally, sparse-coding (introduced via a Laplace prior on the basis functions) [235] is also leveraged to enforce independent basis functions and to simplify the EM algorithm derivation.

The adaptive basis Bayesian Element Free Galerkin introduced in this Chapter provides a clear formulation of Galerkin methods in probabilistic terms. The

formulation can be applied to actual PDE problems (Section 7.5) and suggests directions for future work. A number of simplifications were introduced, namely the vanishing of the first functional derivative and the introduction of sparse basis functions, to derive the model in Section 7.4. These simplifications were made specifically to find a probabilistic interpretation of EFG methods as they are presently applied. The probabilistic formulation, however, suggests that new methods could be developed. For example, Variational Bayesian methods [123] could be used to model the full basis function posterior distribution. Further, it may be possible to bypass the Galerkin simplification entirely and directly improve the estimated solution likelihood function. By investigating these methods, numerical solution procedures for complex problems could potentially be further automated, utilising the power of Machine Learning techniques to find self-improving solutions for PDEs.

7.2 Element Free Galerkin

The Section provides a brief overview of the Element Free Galerkin method for solving Partial Differential Equations in preparation of the adaptive basis method presented in this Chapter. As an overview, this Section does not delve deeply into the formalities of Finite Element Methods, however, the interested reader should consult, for example, the reference list in §9 [198] and [78, 402, 353] for additional details. The basic approach of Galerkin methods is to consider the functional derivatives of the weak-form representation of a PDE problem. By forcing this variation to zero, an error function that is orthogonal to the basis representing the solution function can be minimised. The Element Free Galerkin method, including the Lagrange Multiplier description of boundary conditions, is also introduced based on [33]. After describing the formalism, a parametric representation of basis functions using Artificial Neural Networks and Automatic Differentiation is described.

7.2.1 Basic formulation

Following [353] and §9 of [198], consider the linear Partial Differential Equation:

$$Lu = f \quad x \in \Omega \tag{7.1}$$

for some functions $u(x)$, $f(x)$ over the domain Ω (typically $\Omega \subset \mathbb{R}^d$) and partial differential operator L. This *operational form* can be converted into an integral form more amenable to analysis by considering so-called *weak solutions*.

Following §7 of [310], consider the Sobolev space of functions denoted $\mathbb{H}^m$. For the purposes of this Chapter, it is sufficient to consider an informal definition of these spaces based solely on the $\mathcal{L}^2$ space, however this is a simplification. For a formal definition, see Definition 7.1 of [310]. Using the $\mathcal{L}^2$ simplification of Sobolev spaces, let $\mathbb{H}^m(\Omega)$ be defined as set of functions $u \in \mathcal{L}^2$ over Ω such that the first $\alpha \leq m$ derivatives of u (denoted D^α) are also in $\mathcal{L}^2$. That is, $D^\alpha u \in \mathcal{L}^2$ for $|\alpha| \leq m$. The norm is expressed as the norm of the sum of squares of all u and the derivatives of u:

$$\|u\|_m = \sum_{|\alpha| \leq k} \|D^\alpha u\|_2 \tag{7.2}$$

In the $\mathcal{L}^2$ case considered here, an inner product can be defined by:

$$\langle u, v \rangle = \sum_{\alpha \leq k} \int_\Omega D^\alpha u(x) D^\alpha v(x) dx \tag{7.3}$$

The Galerkin approximation can be derived by introducing two function spaces. Define the unknown solution, u, as belonging to the function space U of *trial functions*. Next, define the space of *test functions*, $v \in V$. For the purposes of this Chapter, it is sufficient to consider $U = V = \mathbb{H}^m$. Following §9 of [198], the weak solution can by introduced by considering the error $G(u) = Lu - f$ for $u \in U$. If, for every $v \in V$, the inner product error functional formed with $G(u)$ is zero:

$$\langle G(u), v \rangle = 0 \tag{7.4}$$

then $u \in U$ is said to be a weak solution of the differential equation. In particular, the error is orthogonal to the trial function space and is thus optimally minimised. The *variational form* of the differential equation $Lu = f$ can be introduced (for positive definite L) following Theorem 9.1 of [198]. The variational form defines an energy functional, $I(u)$, from which the PDE solution, u, can be recovered as the function which renders $I(u)$ stationary. In other words, $Lu = f$ is the Euler-Lagrange equation of $I(u)$. Before demonstrating the proof, it is necessary to place some restrictions on the form of the differential operators L to be analysed. Consider the following properties for

L:

if $\langle Lu, v\rangle = \langle Lv, u\rangle \ \forall u \in U, \forall v \in V$	Self-adjoint
if $\langle Lu, u\rangle > 0 \ \forall u \in U$	Elliptic
if L is both self-adjoint and elliptic	Positive definite

Theorem 7.2.1. Weak-form solution functional (*Theorem 9.1* [198]): Provided the differential operator L is positive definite, $Lu = f$ is the Euler-Lagrange equation of the functional:

$$I(u) = \langle Lu, u\rangle - 2\langle f, u\rangle \quad u \in U \tag{7.5}$$

and, as such, the weak solution of $Lu = f$ is the unique minimum of $I(u)$ for $u \in U$.

Proof. (Proof sketch) By the positive definiteness of L, the variational functional in equation (7.5) possesses a minimum. Let $I(u)$ be the minimum of $I(\circ)$ for $u \in U$. Then consider the first variation of $I(u)$ by $\epsilon \in \mathbb{R}$ (a small value) and $v \in V$:

$$I(u) \leq I(u + \epsilon v) \tag{7.6}$$

where

$$\begin{aligned} I(u + \epsilon v) &= \langle L(u + \epsilon v), (u + \epsilon v)\rangle - 2\langle f, (u + \epsilon v)\rangle \\ I(u + \epsilon v) &= \langle Lu, u\rangle - 2\langle f, u\rangle + \epsilon\left[\langle Lu, v\rangle + \langle Lv, u\rangle - 2\langle f, v\rangle\right] + \epsilon^2\langle Lv, v\rangle \\ I(u + \epsilon v) &= I(u) + 2\epsilon\left[\langle Lu, v\rangle - \langle f, v\rangle\right] + \epsilon^2\langle Lv, v\rangle \end{aligned}$$

so that

$$I(u) \leq I(u + \epsilon v) = I(u) + 2\epsilon\left[\langle Lu, v\rangle - \langle f, v\rangle\right] + \epsilon^2\langle Lv, v\rangle \tag{7.7}$$

Given ϵ and v, the minimum of $I(u)$ occurs if $I(u) = I(u + \epsilon v)$ so that:

$$0 = 2\epsilon\left[\langle Lu, v\rangle - \langle f, v\rangle\right] + \epsilon^2\langle Lv, v\rangle \tag{7.8}$$

Since ϵ can be arbitrarily small and of either sign the term $\epsilon^2\langle Lv, v\rangle$ must

vanish, along with the $[\langle Lu, v\rangle - \langle f, v\rangle]$ term, so that a minimum occurs if:

$$\langle Lu, v\rangle = \langle f, v\rangle \qquad \forall v \in V \tag{7.9}$$

Existence properties required to complete the proof are given in Theorem 9.1 of [198]. □

Note that it is convenient to introduce the following equivalent form of the energy functional in equation (7.5):

$$I(u) = \frac{1}{2}\langle Lu, u\rangle - \langle f, u\rangle \tag{7.10}$$

Additionally, following [246], it is convenient to introduce the standard notation for the Euler-Lagrange equations:

$$a(u, v) = F(u) \tag{7.11}$$

$$\text{where: } a(u, v) = \langle Lu, v\rangle \tag{7.12}$$

$$\text{and: } F(v) = \langle f, v\rangle \tag{7.13}$$

7.2.2 Discretisation of the weak-form

Finally, the Galerkin method introduces an approximation to discretise the weak-form (see §9 of [353]). Specifically, the functions u and v are represented by a basis function expansion:

$$u(x) = \sum_{i=1}^{N} U^i \phi_i(x) \tag{7.14}$$

$$v(x) = \sum_{i=1}^{N} V^i \psi_i(x) \tag{7.15}$$

for $\phi_i(x) \in U$ and $\psi_i(x) \in V$. For the purposes of this Chapter, it is sufficient to consider the case that $U = V$.

The discretised equations can be used to find the weak-form solution. This is a consequence of the Lax-Milgram Theorem (see Theorem 9.2 in [198]).

Expressing equation (7.10), the energy functional $I(u)$, in terms of the discre-

tised u yields a set of matrix equations:

$$I(u) = \frac{1}{2}a\left(\left(\sum_{i=1}^{N} U^i\phi_i\right), \left(\sum_{i=1}^{N} U^i\phi_i\right)\right) - F\left(\sum_{i=1}^{N} U^i\phi_i\right) \tag{7.16}$$

$$I(u) = \frac{1}{2}U^T KU - U^T F \tag{7.17}$$

where U is the column vector of coefficients of $\sum_{i=1}^{N} U^i\phi_i(x)$ with entries U^i, F is the column vector with entries:

$$F_i = \int_\Omega f(x)\phi_i(x)dx \tag{7.18}$$

and K is an $N \times N$ matrix representing the operator L. The integral used to calculate the values of K is typically found by first applying integration by parts to the inner product of $\langle Lu, u\rangle$.

The discrete Euler-Lagrange equations are derived by setting the test functions to be the basis functions in ϕ. This can also be derived by taking the derivative of equation (7.17) with respect to each of the coefficients U^i, which yields (see [78]):

$$\frac{\partial I(u)}{\partial U} = KU - F \tag{7.19}$$

At the minimum value of $I(u)$, the derivative with respect to U should vanish, yielding:

$$KU = F \tag{7.20}$$

For linear PDE problems, equation (7.20) can be solved directly to find the discrete solution function basis coefficients, U.

7.2.3 Boundary conditions

To enforce the boundary conditions of the energy functional, a Lagrange Multiplier approach can be adopted. Alternative approaches, which are not considered further in this Chapter, to enforcing boundary conditions include direct elimination and penalty methods [402]. Following [19, 402], the constrained variational formulation of equation (7.10) can be expressed in terms of the

Lagrangian functional:

$$\mathcal{L}(u, \lambda) = I(u) - \langle \lambda, A(u) - c \rangle \tag{7.21}$$

where $c(x)$ is the assigned value of the solution function, $u(x)$, at $x \in \partial\Omega$ and $\lambda(x)$ is the value of the *Lagrange Multiplier* on the boundary. If u is a stationary point of $I(u)$, then the constraint term will be zero when $A(u(x)) - c(x) = 0$ (where $A(u(x))$ is some boundary condition function) so the stationary point of the Lagrangian at $\mathcal{L}(u, \lambda)$ will also be a solution be a solution of $I(u)$. A more detailed analysis of necessary and sufficient conditions for stationary points of $\mathcal{L}(u, \lambda)$ to provide a solution of $I(u)$ are presented in [198].

As the primary aim of this Chapter is to render clear the probabilistic interpretation of the Galerkin methods, it is useful to restrict the form of the boundary conditions to the most simple type, that is, of the form $A(u(x)) = u(x)$. As such, only Dirichlet boundary conditions are considered in detail (fixed values of the solution function on $\partial\Omega$, the boundary of Ω) are considered in this Chapter. Complicated forms of $A(u(x))$ will render the Lagrangian functional $\mathcal{L}(u, \lambda)$ and its derivatives. The analysis presented will hold as long as the boundary value function $A(u(x))$ does not prevent the existence of a stationary solution $u(x)$ of $\mathcal{L}(u, \lambda)$.

If the solution to the PDE problem is located at a stationary point of $\mathcal{L}(u, \lambda)$, this stationary point can be estimated by considering the first functional variation of $\mathcal{L}(u, \lambda)$ with respect to both u and λ. The derivation of the first variation of $\mathcal{L}(u, \lambda)$ is similar to that presented for the variation of $I(u)$ in Theorem 7.2.1. The first variation of $\mathcal{L}(u, \lambda)$ at a minimum can be calculated by expanding:

$$\mathcal{L}(u, \lambda) \leq \mathcal{L}(u + \epsilon v, \lambda + \delta w)$$

for test functions v, w and scalars ϵ, δ. Expanding these terms yields:

$$\begin{aligned}
\mathcal{L}(u + \epsilon v, \lambda + \delta w) &= I(u + \epsilon v) + \langle \lambda + \delta w, u + \epsilon v - c \rangle \\
I(u + \epsilon v) &= I(u) + 2\left[\langle Lu, v \rangle - \langle f, v \rangle\right] + \epsilon^2 \langle v, v \rangle \\
\langle \lambda + \delta w, u + \epsilon v - c \rangle &= \langle \lambda, u - c \rangle + \epsilon \langle \lambda, v \rangle + \delta \langle w, u \rangle - \delta \langle w, c \rangle \delta \epsilon \langle w, v \rangle
\end{aligned}$$

Setting $\delta = \frac{1}{2}\epsilon$ and combining terms yields:

$$\begin{aligned}\mathcal{L}(u + \epsilon v, \lambda + \delta w) = \mathcal{L}(u, \lambda) + \epsilon^2 \left[\langle Lv, v\rangle + 2\langle w, v\rangle\right] \\ +2\epsilon[\langle Lu, v\rangle - \langle f, v\rangle + \langle \lambda, v\rangle + \langle w, u\rangle - \langle w, c\rangle]\end{aligned}$$

then, there is a stationary point for $\mathcal{L}(u, \lambda)$ when the first variation vanishes:

$$0 = 2\epsilon \left[\langle Lu, v\rangle - \langle f, v\rangle + \langle \lambda, v\rangle + \langle w, u\rangle - \langle w, c\rangle\right] \tag{7.22}$$

Rearranging equation (7.22) yields a set of equations to solve to find the stationary point of $\mathcal{L}(u, \lambda)$:

$$\langle Lu, v\rangle + \langle w, u\rangle + \langle \lambda, v\rangle = \langle f, v\rangle + \langle w, c\rangle \tag{7.23}$$

Following [402], discretising the Lagrange Multiplier equations, using the same technique described in Section 7.2.2, yields the bordered Hessian matrix form of the linear system:

$$\begin{bmatrix} K & A^T \\ A & B \end{bmatrix} \begin{bmatrix} U \\ \Lambda \end{bmatrix} = \begin{bmatrix} F \\ C \end{bmatrix} \tag{7.24}$$

where B is the zero matrix and A is a given by the discretised values of $\langle w, u\rangle$. Λ represents the vector with entries equal to the values of the Lagrange Multipliers required to enforce the constraints. C is a vector with entries:

$$C_i = \langle w_i, c\rangle \tag{7.25}$$

where w_i is the i-th test function for the variation of the boundary constraints.

For the one-dimensional boundary value problem considered in the numerical experiment in Section 7.5, the constraint terms in equation (7.24) can be calculated directly by setting Λ to have two constraints and calculating the entries of A to the values of the basis functions at the bounding end points. Let $\phi_i(x_l)$ be the value of the i-th basis function at the left boundary point, x_l, and $\phi_i(x_r)$ be the value of the i-th basis function at the right end point, x_r. The the first row of A consists of the values of $\phi_i(x_l)$ and the second row

is given by the values of $\phi_i(x_r)$:

$$A_{1i} = \phi_i(x_l) \tag{7.26}$$

$$A_{2i} = \phi_i(x_r) \tag{7.27}$$

Techniques for solving the augmented linear system derived via Lagrange Multiplier constraints are described in [402]. The simplest approach is to augment the diagonal lower zero block in the bordered Hessian as:

$$B = \alpha \mathbf{I} \tag{7.28}$$

where α is a small real valued constant. This is analogous to including a white noise term in a Gaussian Process correlation matrix (see §15 of [275]) and renders the linear system in equation (7.24) invertible. This approach is adopted for the numerical experiment in Section 7.5.

7.2.4 Parametric Representation of basis functions

Artificial Neural Networks can be used to represent both the solution field and the basis functions parametrically. Consider a representation of a scalar solution function, $u(x)$, given by a feedforward ANN (see [173]). The inputs to the ANN are given by the spatial position, x. The output of the ANN is the value of $u_\theta(x)$ where θ represents the network parameters. Then the parameters θ will (along with the choice of ANN architecture) encode how to compute $u(x)$. Such a representation requires that the network is both wide and deep enough to encode the value of the solution function. The second last layer of such a network can be used to represent a series of basis functions, $\Phi_\theta(x)$. The final network weights represent the linear combination of these basis functions, that is:

$$u_\theta(x) = \sum_{i=1}^{N} W_i \Phi_\theta(x) \tag{7.29}$$

where W_i represent the weights used to combine the basis functions. Such an approach could easily be extended to vector-valued solution functions (as required in elastoplastic analysis) by using a matrix of weights W_{ij}, rather than just a vector with entries W_i. Only the scalar case is considered in this Chapter.

Given a training objective, $J(\theta)$, the backpropagation algorithm (see [151]) can be used to update both the basis functions and the weights, W_i. The challenge is to derive such an objective function. Section 7.4 demonstrates that by applying Bayesian Inference principles, an adaptive basis function objective function can be derived via a probabilistic interpretation of Galerkin methods in terms of Bayesian parameter regression.

7.2.5 Discussion

This Section provided a brief overview of Galerkin PDE methods. An ANN based parametric representation of basis functions was introduced. Using an ANN representation, it is possible to adjust the basis functions towards better representations of the PDE solution. However, this Chapter has not yet discussed what training objective should be used to adjust the basis function parameters. Both h and p refinements for FEM improve analysis accuracy based on the a priori known convergence properties of particular function series. By introducing a probabilistic interpretation of constrained optimisation, a gradient-based method for adapting basis functions using backpropagation can be derived. To understand the probabilistic approach, it is necessary to first introduce a number of concepts from probability and Information Theory. This material is the subject of Section 7.3.

7.3 Background theory in Bayesian Inference and optimisation

This Section describes the necessary background material to develop the Bayesian interpretation of the Galerkin method presented in Section 7.2. Bayesian Regression is first described. This provides a probabilistic interpretation of basis function regression for solving PDEs. Next, the Kullback-Liebler (KL) divergence [275] is briefly described for notation consistency. The KL divergence provides a way to measure the distance between two probability distributions. Finally, the Expectation-Maximisation (EM) algorithm is described. This algorithm provides a way to calculate locally optimal MLE/MAP estimates of latent variables for parameterised distributions by iterative updates. The EM algorithm is combined with Galerkin methods to derive the parametric adap-

tive basis function PDE solver presented in this Chapter. This Section also describes the sparse-coding framework [288, 235]. Sparse-coding provides a way to iteratively improve sparse, over-complete basis function sets. In probabilistic terms, sparse-coding introduces latent variable priors that simplify estimates of model likelihood integrals.

7.3.1 Kernelised Bayesian Linear Regression

To understand the Bayesian interpretation of Galerkin methods, it necessary to first outline the Bayesian form of regression. This will be first discussed with reference to the linear regression case, that is, finding the optimal weight vector W such that $f(x) = Wx$. Kernelised regression refers to the use of feature functions, $\phi(x)$, to introduce nonlinearity to regression representations of the form $f(x) = W\phi(x)$. The Bayesian framework represents the unknown weights with distributions over weight space. This is a type of dual representation to the Gaussian Process representation of an unknown function discussed in §15 of [275]. From the Bayesian formulation of regression, the standard Ordinary Least Squares regression estimate can be recovered as a Maximum Likelihood Estimate of the parameter weights.

7.3.2 Bayesian Linear Regression

Bayesian Linear Regression models the probability of the function output given an input stochastically. §3 of [46] provides a detailed overview. The regression weights are taken to be a set of latent variables that are to be learnt. Nonlinearity can be introduced by applying feature functions to the inputs and using the regression weights to build linear combinations in the feature space [268]. The usual least squares estimate can be recovered from the Bayesian linear regression formulation as the Maximum Likelihood Estimate (maximum probability) of the regression weights. Regularised least squares can be justified based on the form of Bayesian linear regression under different prior distribution selections for the regression weights [275]. By adopting the Bayesian linear regression formulation, it will be possible to write an optimisation objective that can be used to iteratively improve the test functions used to represent spectral element solutions of PDEs. This optimisation objective will be shown

to be the model evidence of the estimated solution function given the observations of the PDE solution.

Consider the following model. Let x be a vector in $\mathbb{R}^{D_x}$ and $y \in \mathbb{R}^{D_y}$. Assume that y is a linear function of x with added Gaussian noise such that:

$$y = Wx + \epsilon \tag{7.30}$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2) \tag{7.31}$$

$$P(y|x, W, \sigma^2) \sim \mathcal{N}(Wx, \sigma^2) \tag{7.32}$$

where $W \in \mathbb{R}^{D_x \times D_y}$ is a matrix of latent variables that are to be learnt. For the purposes of this Chapter, it will be sufficient to consider the case that σ is known. For the case that σ must also be inferred, see §7.6.3 of [275].

Consider the set of N training examples:

$$\mathcal{T} = [(x_1, y_1), \cdots, (x_N, y_N)] \tag{7.33}$$

Let X be a matrix of size $N \times D_x$ (the design matrix) and Y be a matrix of size $N \times D_y$. It will be sufficient for the purposes of this Chapter to consider the simpler case that y is a scalar function so that $D_y = 1$. For results regarding Bayesian Multivariate Regression, see [268].

From the properties of Gaussian distributions, the conditional probability of Y given the current values of W is:

$$P(Y|X, W, \sigma^2 \mathbf{I}_N) \propto \exp\left(-\frac{1}{2\sigma^2}\left((Y - XW)^T(Y - XW)\right)\right) \tag{7.34}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix.

The Ordinary Least Squares estimate of regression weights can be recovered as the Maximum Likelihood Estimate for equation (7.34), found by maximising the log of $P(Y|X, W, \sigma^2 \mathbf{I}_N)$ by taking derivatives with respect to the weights:

$$\log P(Y|X, W, \sigma^2 \mathbf{I}_N) \propto \left(-\frac{1}{2\sigma^2}(Y - XW)^T(Y - XW)\right)$$

$$\frac{\partial \log P(Y|X, W, \sigma^2 \mathbf{I}_N)}{\partial W} = X^T XW - X^T Y$$

and setting this derivative to be zero:

$$0 = X^T XW - X^T Y$$
$$W = (X^T X)^{-1} X^T y$$

where $(X^T X)^{-1} X^T$ is the Moore-Penrose Pseudo-Inverse of X [148].

7.3.2.1 Bayesian updating of regression parameters

Rather than use the least squares estimate, a Bayesian approach considers uncertainty over the latent parameter values. Bayesian updating provides a way to incorporate new information about a model, m, given new data, d [224]. The goal is to calculate the posterior distribution, $P(m|d)$, over the models m given new data d using the likelihood, $P(d|m)$, and the prior model beliefs, $P(m)$, by:

$$\underbrace{P(m|d)}_{posterior} \propto \underbrace{P(d|m)}_{likelihood} \underbrace{P(m)}_{prior} \tag{7.35}$$

The Bayesian updating process can be used to continue to update prior beliefs given new data in a rigorous way. Then, in the regression case, the goal will be to estimate the posterior distribution $P(m|d)$ over W given the observations in $\mathcal{T}$:

$$P(W|X, Y, \sigma^2) \tag{7.36}$$

In Bayesian terms, the probability of the observed value y for the regression in equation (7.34) is the likelihood $P(d|m)$. Further, in order to carry out the Bayesian updating for W it is necessary to select a initial prior distribution over W given by $P(W)$. In summary, the Bayesian Linear Regression problem as described is given by:

$$P(d|m) = P(Y|X, W, \sigma^2 \mathbf{I}_N) \quad : \text{Likelihood} \tag{7.37}$$
$$P(m) = P(W) \quad : \text{Prior} \tag{7.38}$$
$$P(m|d) = P(W|X, Y, \sigma^2) \quad : \text{Posterior} \tag{7.39}$$

Although the choice of prior can be essentially arbitrary, the choice of prior affects the convergence of the posterior distribution. For example, consider a

Gaussian prior for $P(W)$ with mean W_0 and covariance matrix V_0:

$$P(W) = \mathcal{N}(W|W_0, V_0) \tag{7.40}$$

The Gaussian prior for $P(W)$ allows for Theorem 3.5.1 to be applied to calculate the posterior update for W given $\mathcal{T}$. Following §7.6.1 of [275], the latent variable posterior update for a Gaussian prior is:

$$P(W|X, Y, \sigma^2) = P(Y|X, W, \sigma^2\mathbf{I}_N)P(W) \tag{7.41}$$
$$P(W|X, Y, \sigma^2) = \mathcal{N}(WX, \sigma^2)\mathcal{N}(W|W_0, V_0) \tag{7.42}$$
$$P(W|X, Y, \sigma^2) = \mathcal{N}(W|W_N, V_N) \tag{7.43}$$

where $P(W|X, Y, \sigma^2) = \mathcal{N}(W|W_N, V_N)$ follows from Theorem 3.5.1. The parameters of the posterior Gaussian are:

$$W_N = V_N V_0^{-1} W_0 + \sigma^{-2} V_N X^T Y \tag{7.44}$$
$$V_N^{-1} = V_0^{-1} + \sigma^{-2} X^T X \tag{7.45}$$
$$V_N = \sigma^2 \left(\sigma^2 V_0^{-1} + X^T X\right)^{-1} \tag{7.46}$$

Importantly, the posterior is also Gaussian. As such, Bayesian updating for incorporating new information will follow the same equations given above, using the current posterior estimate $\mathcal{N}(W|W_N, V_N)$. This can be shown to be equivalent to ridge regression (also known as Tikohonov Regularisation) [373]. In the case that the prior distribution for the weights is non-Gaussian, the form of the posterior will be altered. In Section 7.4, non-Gaussian priors are used to simplify the form of the adaptive basis Galerkin equations.

Kernelised regression can be derived simply by replacing x in equation (7.30) by $\Phi(x)$, where $\Phi(x)$ represents a vector of basis functions evaluated at x:

$$y(x) = W\Phi(x) + \epsilon \tag{7.47}$$

Under the assumption of independent Gaussian noise, ϵ, the probability of any collection of N function values of $y(x)$ at locations x_i for $0, \cdots, i, \cdots, N$ is then:

$$P(y|W, x, \sigma^2, \Phi) = \prod_{i=1}^{N} \mathcal{N}(W\Phi(x_i), \sigma^2) \tag{7.48}$$

This is a form of mean-field approximation to the unknown function value [289]. Note that the Bayesian formulation of regression can also be used to derive variance estimates for the weight posterior, allowing for errors to be quantified. This is discussed in [46, 275].

7.3.3 KL divergence

The Kullback-Liebler divergence provides a measure of the difference between two probability distributions. Errors in the use of approximations like the mean-field approximation in equation (7.48) can be formulated in terms of the KL divergence between the approximation and the true distribution. Minimising these KL divergences yields iterative approximation algorithms such as Expectation Maximisation and Variational Bayes [46].

From §1.6.1 of [46], the KL divergence between two distributions is defined by:

$$D_{KL}(P(x)||Q(x)) = \int P(x) \log \left(\frac{P(x)}{Q(x)} \right) dx \tag{7.49}$$

Importantly, the KL divergence is non-negative so that $D_{KL}(P(x)||Q(x)) \geq 0$ with equality essentially meaning that the distributions $P(x)$ and $Q(x)$ are equivalent. Note that $D_{KL}(P(x)||Q(x)) \neq D_{KL}(Q(x)||P(x))$.

7.3.4 Expectation-Maximisation algorithm

The Expectation-Maximisation algorithm iteratively finds MAP and MLE estimates. The properties of the KL divergence (and entropy) are used to define a two step method to optimise parametric descriptions of latent variable models. For full derivations of the EM algorithm, see §9 of [46]. Note that the EM algorithm is a local optimisation method and is not guaranteed to find global optima.

A parameterised likelihood function describing some observed data, $L(\theta; X) = P(X|\theta)$ can difficult to optimise directly. If latent variables, Z, are assumed to be the cause of the observed data, X, it is often the case that a simpler likelihood function, $L(\theta; X, Z) = P(X, Z|\theta)$, is available. The original likelihood function can be expressed as the marginal likelihood of the integrated

joint observed-latent model:

$$P(X|\theta) = \int_{z \in Z} P(X, z|\theta) dz \tag{7.50}$$

Directly integrating $P(X, Z|\theta)$ to find optimal θ values is often intractable and as such an iterative approach, the EM algorithm, can be adopted. Each iteration of the EM algorithm has two main steps. On iteration i, the Expectation step calculates the expected value with respect to the current value of the conditional distribution $P(Z|X, \theta^i)$, of the log of the joint likelihood function. This gives rise to the so-called auxiliary function, $Q(\theta|\theta^i)$:

$$Q(\theta|\theta^i) = \int_z P(z = Z|X, \theta^i) \log P(X, z = Z|\theta) \tag{7.51}$$

Next, the Maximisation step finds the values of θ^{i+1} that maximise the auxiliary function:

$$\theta^{i+1} = \underset{\theta}{\operatorname{argmax}}\ Q(\theta|\theta^i) \tag{7.52}$$

Repeating the Expectation and Maximisation steps will allow θ to converge to a local MLE/MAP estimate. A typical application of the EM algorthm is fitting the parameters of a Gaussian mixture model [275]. In Section 7.4, an EM algorithm is used to derive an method to iteratively update a parametric representation of the Galerkin basis functions used to solve PDE equations.

7.3.5 Sparse-coding - development and probabilistic interpretation

Consider the problem of learning basis functions for representing a function as:

$$y(x) = \sum_{i=1}^{N} W_i \phi_i(x) \tag{7.53}$$

Sparse-coding, in contrast to PCA methods (see [374]), aims to compute sets of basis functions with a higher dimensionality than the input dimension. Originally derived for estimating useful basis-features for unsupervised learning tasks such as feature extraction from images [235], sparse-coding estimates over-complete basis sets. By removing the limitations of dimensionality that

restrict the eigen-analysis of images, over-complete basis functions allow for effective generalisation from seen to unseen images. This has applications in image classification. The goal of this Section is to briefly introduce sparse-coding in order to discuss the probabilistic interpretation of this method. Sparse-coding introduces a form of regularisation to optimisation algorithms for estimating basis functions. The probabilistic view of this sort of regularisation is that particular forms of priors can be introduced to simplify the task of estimating certain integrals.Sparse-coding priors will be used to simplify the form of the adaptive basis Galerkin EM algorithm presented in this Chapter.

From [288, 235] and §13 of [275] sparse-coding sets the following optimisation problem objective function:

$$\underset{W,\phi}{\operatorname{argmin}} \left[\|y(x_j) - W^j \phi\|_2^2 - \lambda \|\phi\|_1 \right] \tag{7.54}$$

such that the weights, W, are normalised. The parameter λ is a hyperparameter controllng the strength of the $\mathcal{L}^1$ regularisation applied to the basis functions.

This can be solved by a two step approach. First, the current values of the weights, W, are estimated. Next, the basis functions are updated using the current estimate of the weights. Such an optimisation objective can be derived by setting a Laplace prior on the basis functions:

$$P(\phi) \propto \exp(-\beta|\phi|) \tag{7.55}$$

By setting β to a large value, any integral over the space of basis functions will be approximately a delta function centred at a particular value of distribution over the basis functions.

7.4 Bayesian Element Free Galerkin

This Section details the main contribution of this Chapter, the Bayesian formulation of Galerkin methods for solving PDE problems. By unifying probability theory and the solution of PDEs (in the probabilistic numerics sense [175]), directions for finding improved algorithms in the future are suggested.

Further, by applying Bayesian probability techniques it is possible to define parametric adaptive basis Galerkin methods. This Section presents one such approach for adaptive basis Galerkin. A Laplace prior is introduced to simpify the derivation of an EM algorithm for updating basis functions representing the solution of PDEs. A numerical demonstration of this algorithm is presented in Section 7.5. Although the algorithm presented introduces a number of simplifying assumptions that will likely be unsuitable for the solution of complex PDE problems, the derivations in this Chapter clearly identify the probabilistic structure of PDE problems. For example, constrained optimisation for boundary value problems can be related to MAP optimisation for a particular, restrictive prior over the solution space. The goal of this Section is to highlight this probabilistic structure, demonstrate how Galerkin methods fit in with the probabilistic framework and thereby facilitate future developments in parametric adaptive PDE solution methods.

Consider a boundary value problem given by $Lu = f$, as in equation (7.1), over Ω with boundary values $u = c$ over $\partial\Omega$. One method for solving PDE equations via a Bayesian framework would be to attempt to directly parameterise the solution function as, for example, an ANN with latent parameters θ:

$$H(u_\theta) = \int_\Omega \left[Lu_\theta(x) - f(x)\right]^2 dx + \int_{\partial\Omega} \left[u_\theta(x) - c(x)\right]^2 dx \tag{7.56}$$

Taking the value $H(u_\theta)$ as the energy of a Gibbs distribution (see [228]), the likelihood of a set of parameters is given by:

$$P(u_\theta) = \frac{1}{Z} e^{-\beta H(u_\theta)} \tag{7.57}$$

where Z is the partition function over the parameters and β is the inverse temperature tolerance parameter. Such a formulation can be considered be a type of mean field approximation to the true unknown latent solution function,

$u(x)$, and is given by:

$$P(u_\theta) = \frac{1}{Z} e^{-\beta H(u_\theta)} \tag{7.58}$$

$$P(u_\theta) = \frac{1}{Z} e^{-\beta \left[\int_\Omega [Lu_\theta(x) - f(x)]^2 dx + \int_{\partial\Omega} [u_\theta(x) - c(x)]^2 dx\right]} \tag{7.59}$$

$$P(u_\theta) = \frac{1}{Z} e^{-\beta \int_\Omega [Lu_\theta(x) - f(x)]^2 dx} e^{-\beta \int_{\partial\Omega} [u_\theta(x) - c(x)]^2} \tag{7.60}$$

$$P(u_\theta) = \frac{1}{Z} \prod_{i=1}^{\infty} e^{-\beta [Lu_\theta(x_i) - f(x_i)]^2} \prod_{j=1}^{\infty} e^{-\beta [u_\theta(x_j) - c(x_j)]^2} \tag{7.61}$$

Given equation (7.61), finding a solution to the PDE $Lu = f$ by calculating the optimal θ would correspond to a Maximum Likelihood Estimate of θ. There is an interpretation of equation (7.61) as the product of a prior and a likelihood function:

$$P(u_\theta, \hat{c}) = P(u_\theta | \hat{c}) P(\hat{c}) \tag{7.62}$$

$$P(u_\theta | \hat{c}) = \frac{1}{Z} \prod_{i=1}^{\infty} e^{-\beta [Lu_\theta(x_i) - f(x_i)]^2} \prod_{j=1}^{\infty} e^{-\beta [u_\theta(x_j) - \hat{c}(x_j)]^2} \tag{7.63}$$

$$P(\hat{c}) = \delta(\hat{c}(x) - c(x)) \tag{7.64}$$

That is, the prior on the boundary values is a delta prior with $P(\hat{c}) = \delta(\hat{c}(x) - c(x))$ so the probability for the boundary value to be something other than $c(x)$ is zero. From this perspective, the constrained optimisation problems that arise when solving boundary value PDEs can be viewed as MAP estimation problems. The boundary constraints represent priors about particular values of the solution function. Relaxing the prior $\delta(\hat{c}(x) - c(x))$ to, for example, the form of a Laplace prior $\exp(-\beta|\hat{c}(x) - c(x)|)$ would lead to regularised Lagrange Multiplier type boundary constrain implementations.

Although it would be possible to directly optimise equation (7.61), the goal of this Chapter is to demonstrate how Galerkin methods, in particular with reference to the Lagrange Multiplier formulation presented in earlier in this Chapter, can be understood from a Bayesian perspective. Galerkin methods simplify the form the MLE estimate of equation (7.61). Further, the Bayesian Galerkin approach can be used to yield a simple parametric adaptive basis formulation based on sparse priors.

7.4.1 Adaptive Element Free Galerkin by Expectation Maximisation

Galerkin methods can be used to derive a simplified form of the likelihood distribution in equation (7.61). Consider the ANN (described in Section 7.2.4) which represents the value of the solution $u_\theta(x)$ in terms of two subsets of parameters, the weights W and the basis functions $\Phi_\theta(x)$:

$$u_\theta(x) = W\Phi_\theta(x) + \epsilon \tag{7.65}$$

where ϵ is assumed to be independent Gaussian noise with variance σ^2. As in equation (7.1), the likelihood probability of a proposed solution at a collection of points $x = \{x_i\}_{i=1}^N$ is taken to be:

$$P(u_\theta|X, W, \Phi_\theta) = \prod_{i=1}^{N} \mathcal{N}(u_\theta|W\Phi_\theta(x), \sigma^2) \tag{7.66}$$

To optimise the parameters, it will be necessary to derive a likelihood function of the form:

$$P(u_\theta, W, \Phi_\theta|X) = P(u_\theta|X, W, \Phi_\theta)P(W|\Phi_\theta)P(\Phi_\theta) \tag{7.67}$$

From the derivation of Galerkin methods in Section 7.2, the energy due to error in the Lagrangian, $\mathcal{L}(u, \lambda)$, can be taken to be the log probability of $P(W|\Phi_\theta)$. This is the probability of the regression weights given the basis functions. Of course, it is possible to solve the linear system in equation (7.24) by a variety of methods. As per the discussion of Bayesian regression in Section 7.3, the solution of linear systems of equations can be viewed as an MLE of a probabilistic regression problem. By introducing particular prior assumptions on $P(\Phi_\theta)$ and using the Galerkin approximation for $P(W|\Phi_\theta)$, EM can be used to derive an optimisation objective, $J(\theta)$, for the ANN basis model. The optimisation loss objective (to be minimised) is the negative of the auxiliary function (as $Q(\theta|\theta^i)$ is to be maximised) so that:

$$J(\theta) = -Q(\theta|\theta^i) \tag{7.68}$$

Given such a training objective, backpropagation and Stochastic Gradient

Descent (see [151] and Chapter 5) can be used to implement the Maximisation step, finding the optimal values of θ in $Q(\theta|\theta^i)$. This EM algorithm implements an adaptive basis Element Free Galerkin method by iteratively solving the linear system derived from $\mathcal{L}(u_\theta, \lambda)$ given the current basis functions and then training the ANN basis model to more closely model the current solution MLE estimate. The algorithm is derived in detail in this Section.

7.4.2 Derivation of adaptive basis training objective

Consider some distribution over functions, $P(u)$, that describes the true distribution over the solutions for some linear PDE $Lu = f$. Introduce a parameterised distribution, $P(u|\theta)$, intended to approximate the distribution $P(u)$. If the values of θ can be adjusted such that the KL divergence between $P(u)$ and $P(u|\theta)$ is minimised, then $P(u|\theta)$ will be a good approximation of $P(u)$. That is, the goal is to find θ to minimise:

$$D_{KL}\left(P(u)||P(u|\theta)\right) = \int P(u) \log\left(\frac{P(u)}{P(u|\theta)}\right) du \tag{7.69}$$

$$D_{KL}\left(P(u)||P(u|\theta)\right) = \int P(u) \log P(u) du - \int P(u) \log P(u|\theta) du \tag{7.70}$$

As $\int P(u) \log P(u) du$ is a constant and $\log P(u|\theta)$ is always negative:

$$-\int P(u) \log P(u|\theta) du \geq 0 \tag{7.71}$$

The maximum possible value of $\log P(u|\theta)$ is one and, due to the monotonicity of log, the minimum value of $D_{KL}\left(P(u)||P(u|\theta)\right)$ occurs when $\log P(u|\theta)$ is a maximum. That is, optimising the value of the parameters θ to maximise $\log P(u|\theta)$ is equivalent to minimising the KL divergence between $P(u)$ and $P(u|\theta)$. Setting $P(u|\theta) = P(u_\theta)$ in Section (7.4.1) suggests an EM algorithm for maximising $P(u|\theta)$.

7.4.2.1 Expectation-Maximisation for Bayesian Element Free Galerkin

The parameterised distribution $P(u_\theta)$ has a complicated form, given in equation (7.61). Rather than attempt to optimise this directly, a Galerkin method

can be used to introduce the following latent variable model:

$$P(u_\theta) = \int_W \int_\Phi P(u_\theta, w, \phi) dw d\phi \tag{7.72}$$

where the weights w are taken from weight space W and the basis functions are taken from the parameterised set of functions Φ_θ.

The EM algorithm requires an expression for the joint likelihood function $P(X, Z|\theta)$ which, in this case, is taken to be $P(u_\theta, W, \Phi_\theta)$. The log joint likelihood function can be decomposed as:

$$\log P(u_\theta, W, \Phi_\theta) = \log P(u_\theta|W, \Phi_\theta) + \log P(W|\Phi_\theta) + \log P(\Phi_\theta) \tag{7.73}$$

To fully define the Expectation step of EM, the auxiliary function must be estimated. First, consider the required marginal likelihood $P(Z|X, \theta^i)$. For the Galerkin approximation algorithm, this can be given by:

$$P(W, \Phi_{\theta_i}) = P(W, \Phi|\theta^i) \tag{7.74}$$

To estimate $Q(\theta|\theta^i)$, the expectation of the joint likelihood with respect to the marginal likelihood is needed:

$$Q(\theta|\theta^i) = \int_W \int_\Phi P(w, \phi|\theta^{i+1}) \log P(u_\theta, W, \Phi_\theta) d\phi dw \tag{7.75}$$

At this point, there are a number of ways to proceed. In this Chapter, as the intent is to demonstrate the validity of the probabilistic formulation, a Laplace prior is introduced to simplify the calculation of $Q(\theta|\theta^i)$. More sophisticated algorithms are relegated to future work.

7.4.2.2 Sparse-coding simplification over basis function space

The first step to simplifying equation (7.75) is to introduce a Laplace prior over the basis function space:

$$P(\Phi) \propto \exp(-\beta|\hat{\phi}|) \tag{7.76}$$

for some high value of β at some selected initial value $\hat{\phi}$. Then, any terms $P(\Phi)$ appearing in integrals can be approximated as (for some arbitrary variable A):

$$P(A) = \int_{\Phi} P(A, \phi) d\phi \tag{7.77}$$

$$P(A) = \int_{\Phi} P(A|\phi) P(\phi) d\phi \tag{7.78}$$

$$P(A) \approx \int_{\Phi} P(A|\phi) \delta(\hat{\phi} - \phi) d\phi \tag{7.79}$$

$$P(A) \approx P(A|\hat{\phi}) \tag{7.80}$$

Returning to the derivation of $Q(\theta|\theta^i)$, the Laplace prior assumption on $P(\Phi|\theta^i)$ on parameter space yields the simplification:

$$Q(\theta|\theta^i) = \int_W \int_{\Phi} P(w|\phi, \theta^{i+1}) P(\phi|\theta^{i+1}) \log P(u_\theta, W, \Phi_\theta) d\phi dw \tag{7.81}$$

$$Q(\theta|\theta^i) \approx \int_W P(w|\hat{\phi}, \theta^{i+1}) \log P(u_\theta, W, \hat{\phi}, \theta) dw \tag{7.82}$$

Further, introducing a Laplace prior on the values of $P(\Phi|\theta)$ will introduce an $\mathcal{L}^1$ regularisation term into the training objective $Q(\theta|\theta^i)$.

7.4.2.3 $\mathcal{L}^2$ regression weight regularisation

To force sparse basis functions to be learnt, it is necessary to offset any regularisation used to penalise the basis functions by also regularising the regression weights. Otherwise, the weights will grow increasing large to offset the decreasing size of the basis functions. An $\mathcal{L}^2$ for the weights can be derived by considering a Gaussian prior over the weights. This fits in well with the Galerkin approximation adopted for the values of $P(W|\Phi, \theta)$. The energy in the Lagrangian formulation of the boundary value PDE problem (given in equation (7.21) for a given choice of basis can be taken to have the form of the log probability of a Gaussian. Equivalently, the error in the choice of solution weights, W, for the discretised linear system in equation (7.24) can assumed to be the energy function for a Gaussian. Let the constrained linear system derived by a discretised Lagrange Multiplier formulation, approximated with basis functions Φ_{θ^i} be summarised as $\|KW - F\|_2^2$. Then the distribution

describing the likelihood of the weights, W, is given by:

$$P(W|\Phi,\theta^i) = \frac{1}{Z}e^{-\beta\|KW-F\|_2^2} \tag{7.83}$$

The Maximum Likelihood Estimate of $P(W|\Phi,\theta^i)$ will be found at the Ordinary Least Squares estimate of the weights found by solving the system $KW = F$. Denote the least squares estimate of W by $\hat{W}$. At a high value of β, virtually all of the probability mass of $P(W|\Phi,\theta^i)$ will be centred at $\hat{W}$. Then, similarly to the simplification introduced in Section 7.4.2.2, the auxiliary function integral in equation (7.82) can be further simplified by:

$$Q(\theta|\theta^i) \approx \int_W P(w|\hat{\phi},\theta^{i+1}) \log P(u_\theta, W, \hat{\phi}, \theta) dw \tag{7.84}$$

$$Q(\theta|\theta^i) \approx \log P(u_\theta, \hat{W}, \hat{\phi}, \theta) \tag{7.85}$$

7.4.2.4 The auxiliary function training objective

Equation (7.85) can be used to derive a training objective for the ANN describing the PDE solutions. Via the EM algorithm, maximising θ in $Q(\theta|\theta^i)$ (the Maximisation step) will maximise the likelihood of the parameters. From the discussion in Section 7.4.2, this will in turn reduce the KL divergence between the approximation $P(u|\theta)$ and the unknown true distribution $P(u)$. To find the training objective, equation (7.85) can be expanded by factorising $P(u_\theta, \hat{W}, \hat{\phi}, \theta)$ into:

$$P(u_\theta, \hat{W}, \hat{\phi}, \theta) = P(u_\theta|\hat{W}, \hat{\phi}, \theta)P(\hat{W}|\hat{\phi}, \theta)P(\hat{\phi}, \theta) \tag{7.86}$$

First, assume that the probability distribution $P(u_\theta|\hat{W}, \hat{\phi}, \theta)$ represents the output of the ANN so that for a set of inputs $x = \{x_i\}$, the probability of a proposed solution is given by:

$$P(u_\theta|\hat{W}, \hat{\phi}, \theta) = \prod_{i=1}^{N} \mathcal{N}\left(u|\hat{W}\hat{\phi}, \sigma^2\right) \tag{7.87}$$

where σ^2 is a noise term which will not be used for the purposes of the algorithm derived in this Section.

To maximise θ, the proposed solution $u_\theta(x)$ from the network should be ad-

justed (by gradient descent) to match the current prediction, $\hat{W}\hat{\phi}(x)$:

$$\log P(u_\theta|x) \propto -\|u_\theta(x) - \hat{W}\hat{\phi}(x)\|_2^2 \tag{7.88}$$

Following the detailed description in §13 of [275], the term $P(\hat{W}|\hat{\phi}, \theta)$ will induce an $\mathcal{L}^2$ regularisation on the on the ANN weights, θ. This can be implemented efficiently by considering weight decay regularisation on only the weights, W_θ^{i+1}, in the final layer of the ANN:

$$\log P(\hat{W}|\hat{\phi}, \theta) \propto -\lambda_W \|W_\theta^{i+1}\|_2^2 \tag{7.89}$$

where λ_W is a hyperparameter describing the strength of the regulariser.

Finally, following §13 of [275], the Laplace prior on the basis functions induces an $\mathcal{L}^1$ regulariser in $Q(\theta|\theta^i)$:

$$\log P(\hat{\phi}, \theta) \propto -\lambda_\Phi \|\phi(x)\|_1 \tag{7.90}$$

where λ_Φ is a hyperparameter describing the strength of the regulariser.

From equation (7.68), the training loss for the ANN can be considered to be the negative of the auxiliary function, $J(\theta) = -Q(\theta|\theta^i)$. Training an ANN to minimise $J(\theta)$ will be equivalent to maximising $Q(\theta|\theta^i)$ which will, in turn, be equivalent to minimising the KL divergence between $P(u)$ and $P(u|\theta)$. The full expression for $J(\theta)$ is given by:

$$J(\theta) = \|u_\theta(x) - \hat{W}\hat{\phi}(x)\|_2^2 + \lambda_W \|W_\theta^{i+1}\|_2^2 + \lambda_\Phi \|\phi(x)\|_1 \tag{7.91}$$

Note that the training objective is integrated over all of $x \in \Omega$. Further, note that the hyperparameters λ_W, λ_Φ must be selected (or also optimised) as a part of the ANN fitting process.

Training an ANN to minimise $J(\theta)$ using the current MLE of the unknown solution function, calculated by a solving the augmented linear system in equation (7.24), will find new basis functions that will increase the likelihood of the estimated solutions.

7.4.3 Summary of algorithm

The sparse-coding Bayesian Element Free Galerkin Method can be summarised as follows. Create a feedforward ANN $u_\theta(x)$ that takes a location, x, as input and produces a scalar variable $u_\theta(x)$ by:

$$u_\theta(x) = W_\theta \Phi_\theta(x) \tag{7.92}$$

Let θ denote all weights in the ANN. The weights W_θ are a subset of θ and are taken to be the regression weights used to calculate the final scalar value of the network given the outputs of the second last layer, denoted $\Phi_\theta(x)$. The second last layer, $\Phi_\theta(x)$, is taken to be a vector in $\mathbb{R}^k$ where k is the number of basis features.

Given a PDE $Lu = f$ and boundary conditions, consider the Lagrangian linear system discretised form of $Lu = f$ given by equation (7.24). Using the current basis set, $\Phi_\theta(x)$, generate the linear system as per the description in Section 7.2.3. Automatic Differentiation can be used to calculate the values of $\nabla_x \Phi(x)$. In this Chapter, Monte Carlo Simulation is used to approximate the values of the required inner product integrals. These Monte Carlo integrals can be carried out by repeatedly sampling uniformly from the input space and then taking the sample average values of the vectors and matrices in equation (7.24).

Next, for the current basis set and equations $KW = F$, calculate a best estimate for the weights, $\hat{W}$. Using the basis functions, the ANN can be trained to reduce the loss function from equation (7.91):

$$J(\theta) = \int_\Omega \|u_\theta(x) - \hat{W}\hat{\phi}(x)\|_2^2 + \lambda_W \|W_\theta^{i+1}\|_2^2 + \lambda_\Phi \|\phi(x)\|_1 dx \tag{7.93}$$

where $\hat{W}\Phi_{\theta^i}(x)$ is the current estimate of the solution with the basis functions used to estimate $\hat{W}$.

Using Stochastic Gradient Descent, the loss function $J(\theta)$, can be approximated by taking N evenly spaced input locations, x_i, over the solution domain in order to generate minibatches (see [151] and Chapter 5). The loss function

can then be approximated as:

$$J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \|u_\theta(x_i) - \hat{W}\hat{\phi}(x_i)\|_2^2 + \lambda_W \|W_\theta^{i+1}\|_2^2 + \lambda_\Phi \|\phi(x_i)\|_1 \qquad (7.94)$$

After training the ANN using the current estimate of the solution function, the process can be repeated. In summary, the algorithm is as follows:

1. Generate an ANN as in equation (7.29).
2. Using the current ANN basis functions, $\Phi_\theta(x)$, generate the linear system $KW = F$ as in equation (7.24). Monte Carlo Integration or some other method can be used to estimate the required integrals.
3. Solve $KW = F$ for the weights $\hat{W}$.
4. Generate an approximation of the solution function at N points, $\{x_i\}_{i=1}^N$, using the current basis functions, $\Phi_\theta(x)$ and the weights, $\hat{W}$, by $\hat{W}\Phi_\theta(x)$.
5. Using the current values of the solution function, $\{\hat{W}\Phi_\theta(x_i)\}_{i=1}^N$, use minibatch Stochastic Gradient Descent to train the ANN to minimise the loss function in equation (7.94).
6. Return to step two and repeat until convergence.

7.4.4 Discussion

This Section presented a formulation of the solution of PDE equations in terms of Bayesian Inference. In particular, it was demonstrated that Galerkin methods can be interpreted as a type of Maximum Likelihood Estimate based on projection of the PDE solution onto latent variable basis functions. This probabilistic formulation is useful for a number of reasons. Galerkin methods can be unified into the framework of probabilistic numerics along with other numerical optimisation methods (see [175]). Adaptive basis FEM and element free methods can be understood in terms of the model likelihood, providing for a rigorous way to understand the quality of different basis function choices. Further, by adopting a Bayesian formulation, the constrained optimisation problems arising in the solution of PDEs can be understood as MAP estimates. Although a Galerkin method for solving constrained PDE problems using Lagrange Multipliers was discussed, viewing constrained optimisation

as a MAP problem suggests alternative procedures. In particular, the EM algorithm (discussed in reference to improving basis functions in this Section) could be used for constrained optimisation or for interpreting existing methods for constrained optimisation. Finally, MLE and MAP estimates could be replaced with Variational Bayesian methods to approximate the required posteriors, avoiding the need to explicitly introduce Galerkin methods to find PDE solutions.

A sparse-coding based EM algorithm for improving the current set of basis functions was discussed in this Section. This algorithm is not intended to be an optimal algorithm for solving all PDE problems. Rather, by adopting sparse (Laplace) priors the form of the EM algorithm can be simplified. The numerical example in Section 7.5 applies the sparse-coding EM algorithm to a simple PDE problem to demonstrate that the probabilistic formulation of solving PDEs can be of practical, as well as theoretical, use. The Laplace prior is, however, too restrictive for complicated problems. By using a prior that places significant probability mass over only a small region of the latent variable space, a Bayesian updating process will be slow (or unable to) converge. As EM finds local, rather than global, optima the poor prior selection problem is likely to be exacerbated when solving more complicated PDEs. Although more advanced algorithms that could deal with these issues are not detailed in this Chapter, by demonstrating the probabilistic structure of PDE solutions this research suggests future directions in adaptive PDE algorithms based on Bayesian methods. Specifically, Variational Bayesian methods and more careful prior selection could be used to derive improved algorithms. The material presented in this Chapter will help to derive these algorithms as a part of future work.

7.5 Numerical example - one dimensional Poisson equation

To demonstrate the validity of this probabilistic interpretation of Galerkin methods, this Section analyses a simple one dimensional Poisson problem. The sparse-coding basis function EM optimisation procedure described in Section 7.4 is an extension of the more fundamental Bayesian interpretation of PDE solvers and Galerkin methods described in this Chapter. Rather than attempt

to solve a complicated problem with a potentially suboptimal algorithm, it is worthwhile to demonstrate the theory with a simple problem. The numerical analysis in this Section suggests that the Bayesian approach to the solution of PDEs is likely to be a viable avenue for future research. Note that one-dimensional PDE problems are also termed Ordinary Differential Equations (ODE). ODEs can be considered to be a subset of PDEs and so the usage of PDE is not inaccurate when describing one-dimensional differential equation problems.

7.5.1 Problem description

Consider the Poisson equation (see [246]):

$$\nabla^2 u(x) = f(x) \qquad x \in \Omega \tag{7.95}$$

$$u(x) = c(x) \qquad x \in \partial\Omega \tag{7.96}$$

$$f(x) = -1 \tag{7.97}$$

$$c(x) = 0 \tag{7.98}$$

where $u(x)$ is the unknown solution function, $f(x)$ is the given source field function and $c(x)$ is a Dirichlet boundary condition. Further, Ω is the spatial domain, given by $[0, 1] \in \mathbb{R}$ and $\partial\Omega$ is the boundary of $[0, 1]$.

The analytical solution to this problem can be solved trivially by taking the integral of equation (7.95) and solving for the integration constants using the boundary conditions. The analytical solution for equation (7.95) is given by:

$$u(x) = -\frac{1}{2}x^2 + \frac{1}{2}x \tag{7.99}$$

The goal of this Section is to validate the algorithm described in Section 7.4.3 on the problem given in equation (7.95). The analytical solution in equation (7.99) is used as a means of verifying the solutions given by the algorithm method proposed in this Chapter.

7.5.2 Analysis details

To implement the sparse-coding EM adaptive basis algorithm, a four layer Artificial Neural Network was used. The ANN was comprised of two layers of

densely connected ELU 50 units (see [77]), followed by a 50 unit wide layer of densely connected tanh units. The final layer output, $u_\theta(x)$, was used to represent the solution function, $u(x)$, in terms of the parametric basis functions, $\Phi_\theta(x)$, given by the outputs of the second last layer. 10 basis functions were used to represent the solution, so $\Phi_\theta(x)$ is a vector in $\mathbb{R}^{10}$ for each value of x. The $\mathcal{L}^2$ regularisation in equation (7.94) was applied to the weights, W_θ, connecting $\Phi_\theta(x)$ and the output so that $u_\theta(x) = W_\theta \Phi_\theta(x)$.

Importantly, the bias weights in the second last layer were disabled (as such a bias basis term was not used to formulate the Euler-Lagrange equations, $KW = F$). The integrals required to formulate the terms in $KW = F$ were estimated by Monte Carlo Integration using 10000 random samples from Ω. The $KW = F$ equations were solved using the SciPy linear algebra implementations [204].

The ANN, as well as the ANN training algorithms, were generated using the Python Tensorflow implementations [1]. The ANN was trained using the Adam optimiser algorithm [212]. A learning rate of 0.001 was adopted for all analyses. Each training iteration lasted for 1000 epochs (gradient descent iterations). Minibatches were calculated using the inputs from 20 evenly spaced points on $[0, 1]$. Regularisation parameters $\lambda_W = 0.01$ and $\lambda_\Phi = 0.01$ were used for all analyses. These training parameters were found experimentally. Automating the search for effective training regimes is an ongoing area of research [151].

7.5.3 Numerical results

The numerical results presented in this Section indicate that the algorithm described in this Chapter successfully solved the Poisson problem in equation (7.95). Figure 7.1 demonstrates the convergence of the approximated solutions, $u_\theta(x)$, over ten iterations of the EM algorithm, as well as the analytical solution. Solutions for iteration 1 and then all even number iterations from 2 to 10 (inclusive) are plotted. Successive approximations to the true solution are shown as darkening lines, that is, earlier solutions are shown as lighter lines. The analytical solution is shown as a dashed line. Figure 7.1 indicates that the algorithm improves upon the approximated solution until the point that the errors due to the Monte Carlo inner product approximations prevent

further convergence.

Figure 7.2 demonstrates the over-complete basis functions learnt by the approximation algorithm at the end of the 10-th iteration of the EM algorithm. Note that the basis functions are not sorted, as would be the case in a PCA type algorithm [374]. The dominant basis functions are essentially the same shape as the analytic solution. The basis functions that do not perfectly track the shape of the solution function are likely capturing the noise present due to numerical approximations of the solution function, in particular due to the MCS integrals used to build the linear system solved at each iteration.

Figure 7.3 demonstrates the errors in the proposed solution versus number of EM algorithm iterations. The errors, $E(j)$ at iteration j, are calculated as the average of the squared difference from the analytic solution:

$$E(j) = \frac{1}{N} \sum_{i=1} (u_\theta(x_i) - u(x_i))^2 \tag{7.100}$$

where the approximation points, x_i, are taken from $N = 100$ evenly spaced points over $\Omega = [0, 1]$. Figure 7.3 demonstrates that the approximation error decreases rapidly (super-logarithmically) but then, around the fifth or sixth iteration, begins to reach the maximum accuracy that can be obtained with the number of MCS iterations used. This suggests that it would be advisable to run the EM algorithm by first using a small number of MCS iterations to build the required linear systems. Then, after several EM iterations have been used to find a good approximation to the basis functions, the number of MCS iterations could be increased. This could be interpreted as a form of simulated annealing (see [305]).

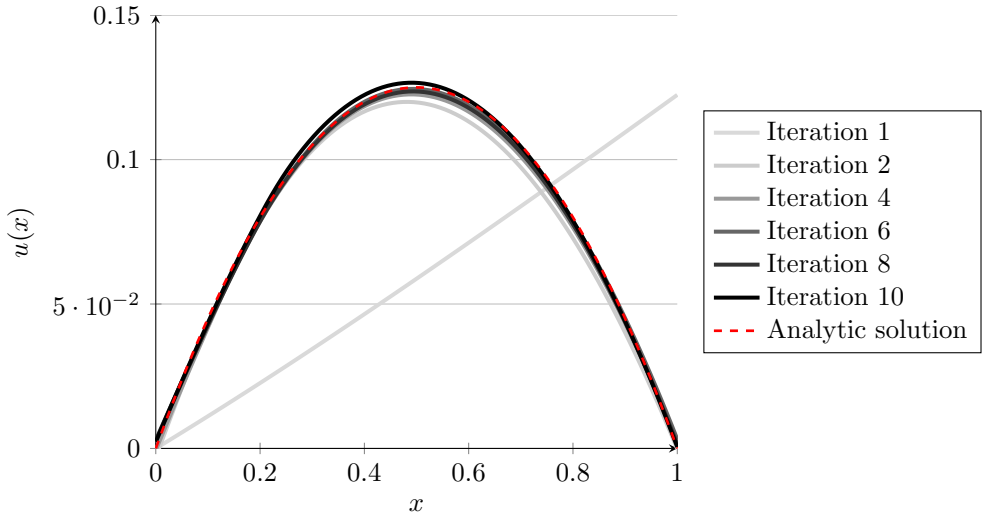


Figure 7.1: Bayesian Element Free Galerkin Poisson problem from equation (7.95) solution convergence. The analytic solution from equation (7.99) is shown as a dashed line. Solutions from the Artificial Neural Network, $u_\theta(x)$, are shown for the initial values and even numbered iterations from 2 to 10 (inclusive).

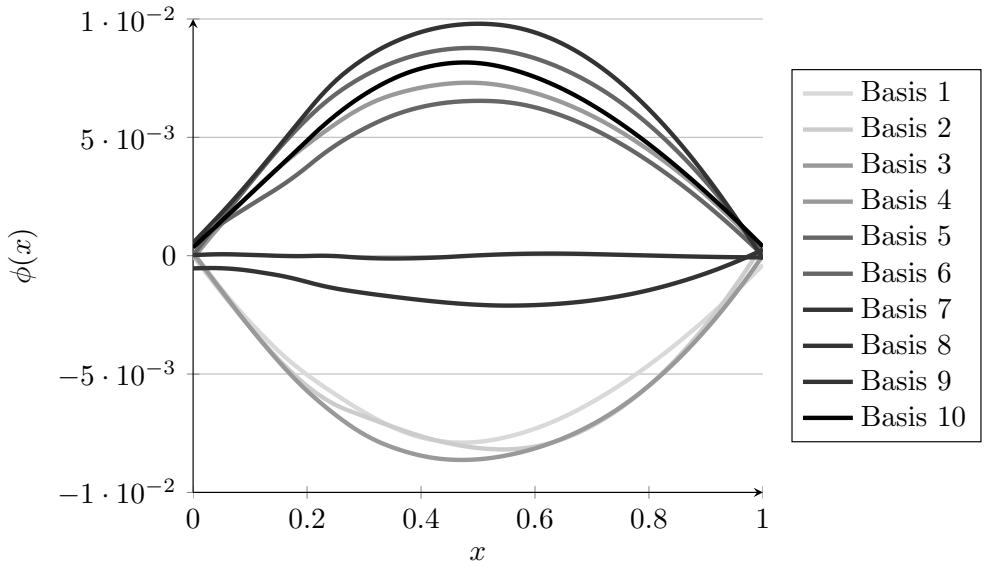


Figure 7.2: Basis functions learnt by Expectation-Maximisation adaptive basis algorithm. Ten basis function features were encoded in the Artificial Neural Network used to approximate the solution, as per the algorithm in Section 7.4. Note that the basis functions are not in any particular order.

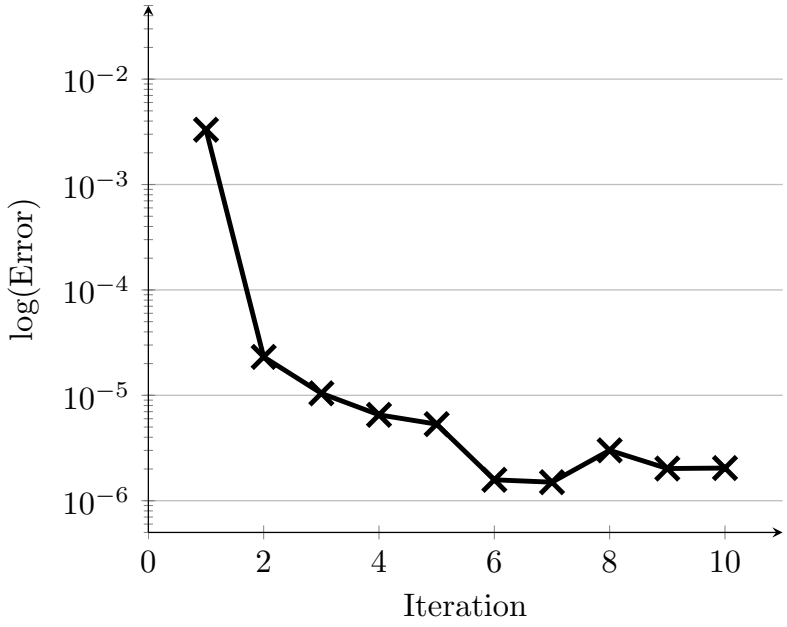


Figure 7.3: Bayesian Element Free Galerkin convergence analysis. Errors are calculated as per equation (7.100) and show the average squared difference between the Artificial Neural Network encoded solution, $u_\theta(x)$, and the analytic solution from equation (7.99). Errors are shown per number of iterations of the Expectation-Maximisation algorithm described in Section 7.4. Note that the vertical axis is logarithmic.

7.6 Conclusions

This Chapter demonstrated a Bayesian interpretation of the solution of differential equations. As well as being of theoretical interest, the Bayesian perspective was used to formulate an adaptive basis element free Galerkin method based on an Expectation-Maximisation algorithm. This was verified on a simple numerical example problem. Although the algorithm presented is unlikely to be optimal for more complex PDE problems, by demonstrating the unifying probabilistic structure of the solution of PDE problems (in particular Galerkin approximations), directions for future research are revealed. By viewing the unknown solution function as set of hidden random variables, techniques from statistical analysis and Machine Learning can be applied to the solution of PDE problems. Bayesian probability is the essential aspect which unlocks the power of probabilistic numerical analysis.

Although in this Chapter, only a one dimensional Poisson equation was investigated, it is encouraging that the technique proposed was effective. Moreover, the 'probabilistic numerics' approach to the analysis of the chosen PDE was useful in that the interpretation of the solution of the analysed PDE by a Galerkin method was made clear. Through a Bayesian lens it was clear that the Galerkin method defines the conditional probability of the regression weights of a proposed solution given a set of basis functions. Although the one dimensional problem tested does not provide a true test of the proposed adaptive basis method, the real contribution of this Chapter is the development of the probabilistic interpretation of numerical methods. It is hoped that this will lead to improved methods, beyond simple Galerkin techniques, with additional research.

From the perspective of Uncertainty Quantification, the ability to efficiently solve PDE problems is essential to estimating unknown output probability densities. Automating the solution of PDE problems by introducing probabilistic models of PDE solvers will facilitate the search for algorithms that can deal with the computational difficulties posed by problems in Uncertainty Quantification. It is envisaged that merging Uncertainty Quantification techniques with automated PDE solvers derived using probabilistic numerical methods will be both necessary and beneficial for dealing with increasingly complicated numerical analysis problems in the future.

Chapter 8

Conclusions

This thesis addressed Uncertainty Quantification for PDE models of physical systems from a number of perspectives. Although traditional methods to Uncertainty Quantification can (as demonstrated in this thesis) be applied very successfully to complex numerical analyses, they face numerous computational difficulties. For example, it was shown that rare event reliability problems can be addressed by specialised sampling techniques. However, it was also demonstrated that numerical methods for physical systems can be computationally intensive for deterministic problems. Probabilistic models require that a number of inputs to a given numerical method are analysed. As well as increasing the computational burden, this presents conceptual difficulties. The space of probabilistic inputs is typically very high dimensional and humans are not well equipped to deal with such spaces intuitively. As such, it is difficult to know exactly what algorithm to pick to deal with a complicated Uncertainty Quantification problem.

From a Machine Learning perspective, the solution to the problem of deriving difficult algorithms is to use an optimisation procedure over the space of algorithms. This thesis made a number of steps in this direction, but more work will be necessary in order to reach fully automated Uncertainty Quantification methods that can be applied to the types of complicated numerical problems encountered in practice. It is the authors point of view that Bayesian (or generative) models of numerical methods will be a useful approach. The theoretical structure of these Bayesian models were explored in this thesis. A probabilistic description of numerical method solvers allows for a unified

interpretation of Uncertainty Quantification. The goal then becomes to quantify the joint input-output and model distributions. This is supported by the work of [175] and by the adaptive Galerkin method presented in this thesis. By interpreting the residual as a Gibbs distribution Hamiltonian, the probabilistic nature of numerical simulation solver models is revealed. The most significant realisation to take away is that, from a probabilistic perspective, the unknown solution to a numerical method can be considered to be a latent variable. Following this logic, the goal of an algorithm is to identify the value of these latent variables given observations. This has an interesting interpretation. From a Variational Bayes perspective, a latent variable posterior can be approximated by reducing the KL divergence of the latent variable model from the estimated distribution of observed data. An optimal algorithm for some problem would then be one that reduces KL divergence between the latent variable posterior and the true, unknown solution as rapidly as possible. This would require the cross entropy between the unknown solution and the solution model would need to be decreased as rapidly as possible. In other words, an optimal algorithm would be one that maximises the rate of entropy production. This analysis applies to all numerical methods, not just those for Uncertainty Quantification. Self-improving algorithms, based on these unifying theoretical developments, are likely to be a fruitful avenue for both theoretical and practical advances in algorithms for a variety of problems in the future.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] A.J. Abbo, A.V. Lyamin, S.W. Sloan, and J.P. Hambleton. A C2 continuous approximation to the Mohr–Coulomb yield surface. *International Journal of Solids and Structures*, 48(21):3001–3010, Oct 2011.

[3] P. Abrahamsen. *A review of Gaussian random fields and correlation functions*. Norsk Regnesentral/Norwegian Computing Center, 1997.

[4] R.J. Adler. *The Geometry of Random Fields (Reprint of 1980 Edition)*. Society for Industrial and Applied Mathematics, Jan 2010.

[5] R.J. Adler and J.E. Taylor. *Random Fields and Geometry (Springer Monographs in Mathematics)*. Springer, 2007.

[6] C.C. Aggarwal. Towards meaningful high-dimensional nearest neighbor search by human-computer interaction. *Proceedings 18th International Conference on Data Engineering*, 2002.

[7] R. Ak, V. Vitelli, and E. Zio. An interval-valued neural network approach for uncertainty quantification in short-term wind speed prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 26(11):2787–2800, Nov 2015.

[8] J. Aldrich. R.A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, Sep 1997.

[9] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M.E. Rognes, and G.N. Wells. The fenics project version 1.5. *Archive of Numerical Software*, 3(100):9–23, 2015.

[10] N.S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175, Aug 1992.

[11] J.E. Angus. The probability integral transform and related results. *SIAM review*, 36(4):652–654, 1994.

[12] D.W. Apley, J. Liu, and W. Chen. Understanding the effects of model uncertainty in robust design with computer experiments. *Journal of Mechanical Design*, 128(4):945, 2006.

[13] S. Asmussen and P.W. Glynn. *Stochastic Simulation: Algorithms and Analysis*. Stochastic Modelling and Applied Probability. Springer New York, 2007.

[14] R.J. Atkin and N. Fox. *An Introduction to the Theory of Elasticity*. Dover Books on Physics. Dover Publications, 2013.

[15] S.-K. Au and J. L. Beck. Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic Engineering Mechanics*, 16(4):263–277, Oct 2001.

[16] S.-K. Au and Y. Wang. *Engineering Risk Assessment with Subset Simulation*. Wiley-Blackwell, Apr 2014.

[17] E. Ayyildiz, V.P. Gazi, and E. Wit. A Short Note on Resolving Singularity Problems in Covariance Matrices. *International Journal of Statistics and Probability*, 1(2):113–118, Sep 2012.

[18] A. Azzalini and A.D. Valle. The multivariate skew-normal distribution. *Biometrika*, 83(4):715–726, 1996.

[19] I. Babuška. The finite element method with lagrangian multipliers. *Numerische Mathematik*, 20(3):179–192, 1973.

[20] I. Babuvška and W.C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.

[21] G.B. Baecher and J.T. Christian. *Reliability and statistics in geotechnical engineering*. Wiley, 2003.

[22] G. Bakir and Neural Information Processing Systems Foundation. *Predicting Structured Data*. Advances in neural information processing systems. MIT Press, 2007.

[23] B. Bakker. Reinforcement learning with long short-term memory. In *In NIPS*, pages 1475–1482. MIT Press, 2002.

[24] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, K. Rupp, B.F. Smith, S. Zampini, and H. Zhang. PETSc Users Manual. Technical Report ANL-95/11 - Revision 3.6, Argonne National Laboratory, 2015.

[25] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, Brune. P., K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, K. Rupp, B.F. Smith, S. Zampini, and H. Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2015.

[26] S. Balay, W.D. Gropp, L.C. McInnes, and B.F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[27] R.C. Batra. *Elements of Continuum Mechanics*. AIAA education series. American Institute of Aeronautics & Astronautics, 2006.

[28] J.A. Beachy and W.D. Blair. *Abstract algebra*. Waveland Press, 2006.

[29] J.L. Beck and S.-K. Au. Bayesian updating of structural models and reliability using Markov chain Monte Carlo simulation. *Journal of Engineering Mechanics*, 128(4):380–391, Apr 2002.

[30] T. Bedford and R. Cooke. *Probabilistic Risk Analysis: Foundations and Methods*. Cambridge University Press, 2001.

[31] J.S. Bell. *Speakable and unspeakable in quantum mechanics: Collected papers on quantum philosophy*. Cambridge University Press, 2004.

[32] R.E. Bellman and S.E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Jan 1962.

[33] T. Belytschko, Y.Y. Lu, and L. Gu. Element-free galerkin methods. *International journal for numerical methods in engineering*, 37(2):229–256, 1994.

[34] A. Ben-Naim. *A Farewell to Entropy: Statistical Thermodynamics Based on Information: S*. World Scientific, 2008.

[35] J.J. Benedetto. *Harmonic Analysis and Applications*. Studies in Advanced Mathematics. Taylor & Francis, 1996.

[36] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade*, pages 437–478, 2012.

[37] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug 2013.

[38] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[39] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, Mar 1994.

[40] M. Bensi, A. Der Kiureghian, and D. Straub. Bayesian network modeling of correlated random variables drawn from a Gaussian random field. *Structural Safety*, 33(6):317 – 332, Sep 2011.

[41] D.P. Bertsekas and W. Rheinboldt. *Constrained Optimization and Lagrange Multiplier Methods*. Computer science and applied mathematics. Elsevier Science, 2014.

[42] M. Berveiller, B. Sudret, and M. Lemaire. Stochastic finite element: a non intrusive approach by regression. *Revue européenne de mécanique numérique*, 15(1-2-3):81–92, Mar 2006.

[43] W. Betz, I. Papaioannou, and D. Straub. Numerical methods for the discretization of random fields by means of the Karhunen-Loève expansion. *Computer Methods in Applied Mechanics and Engineering*, 271:109–129, Apr 2014.

[44] C.M. Bishop. Neural networks and their applications. *Review of Scientific Instruments*, 65(6):1803–1832, Jun 1994.

[45] C.M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[46] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[47] G. Blatman and B. Sudret. An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Probabilistic Engineering Mechanics*, 25(2):183–197, Apr 2010.

[48] G. Blatman and B. Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, 230(6):2345–2367, Mar 2011.

[49] G. Blatman and B. Sudret. Sparse polynomial chaos expansions of vector-valued response quantities. *Safety, Reliability, Risk and Life-Cycle Performance of Structures and Infrastructures*, pages 3245–3252, Jan 2014.

[50] V.I. Bogachev. *Measure Theory*. Springer Nature, 2007.

[51] L. Bottou. Large-scale machine learning with stochastic gradient descent. *Statistical Learning and Data Science*, pages 17–25, Dec 2011.

[52] M. Branicki and A.J. Majda. Fundamental limitations of polynomial chaos for uncertainty quantification in systems with intermittent instabilities. *Communications in Mathematical Sciences*, 11(1):55–103, 2013.

[53] C.E. Brennen. *Fundamentals of multiphase flow*. Cambridge University Press, 2005.

[54] S.C. Brenner and C. Carstensen. Finite Element Methods. *Encyclopedia of Computational Mechanics*, Nov 2004.

[55] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, May 2011.

[56] M.C. Cacas, E. Ledoux, G. de Marsily, B. Tillie, A. Barbreau, E. Durand, B. Feuga, and P. Peaudecerf. Modeling fracture flow with a stochastic discrete fracture network: calibration and validation: 1. the flow model. *Water Resources Research*, 26(3):479–489, 1990.

[57] R.E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *ANU*, 7:1, Jan 1998.

[58] G. Cai and Y. Lin. Generation of non-Gaussian stationary stochastic processes. *Phys. Rev. E*, 54(1):299–303, Jul 1996.

[59] L.L. Cam. Maximum likelihood: An introduction. *International Statistical Review / Revue Internationale de Statistique*, 58(2):153, Aug 1990.

[60] W. Cao, M. Liu, and Z. Fan. Ms-stability of the euler–maruyama method for stochastic differential delay equations. *Applied Mathematics and Computation*, 159(1):127–135, 2004.

[61] G. Casella and R.L. Berger. *Statistical Inference*. Duxbury advanced series. Duxbury Thomson Learning, 2nd edition, 2002.

[62] P. Chadwick. *Continuum mechanics: concise theory and problems*. Courier Dover Publications, 2012.

[63] G.J. Chaitin. Information-theoretic limitations of formal systems. *Journal of the ACM (JACM)*, 21(3):403–424, 1974.

[64] G.J. Chaitin. Algorithmic information theory. *IBM journal of research and development*, 21(4):350–359, 1977.

[65] X.W. Chang, C.C. Paige, and G.W. Stewart. New perturbation analyses for the Cholesky factorization. *IMA Journal Numererical Analysis*, 16(4):457–484, 1996.

[66] X.W. Chang and D. Stehlé. Rigorous Perturbation Bounds of Some Matrix Factorizations. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2841–2859, Jan 2010.

[67] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin. Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research*, 11(Apr):1471–1490, 2010.

[68] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised Learning*. Adaptive Computation and Machi. MIT Press, 2010.

[69] S. Chen and G.D. Doolen. Lattice boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.

[70] W.F. Chen. *Limit analysis and soil plasticity*. Elsevier, 1975.

[71] C. Cherubini. Reliability evaluation of shallow foundation bearing capacity on c' ϕ' soils. *Canadian Geotechnical Journal*, 37(1):264–269, feb 2000.

[72] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327, nov 1995.

[73] D. Child. *The Essentials of Factor Analysis*. Bloomsbury Academic, 2006.

[74] L.N. Childs. *A concrete introduction to higher algebra*. Springer, 2009.

[75] Y. Cho and L.K. Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350, 2009.

[76] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[77] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations 2016*, 2015.

[78] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt. *Concepts and applications of finite element analysis*. John Wiley & Sons, 2007.

[79] M. Cord and P. Cunningham, editors. *Machine Learning Techniques for Multimedia*. Springer Berlin Heidelberg, 2008.

[80] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2009.

[81] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[82] R. Courant and D. Hilbert. *Methods of Mathematical Physics: Partial Differential Equations*, volume 2 of *Wiley Classics Library*. Wiley, 2008.

[83] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley-Blackwell, Apr 2005.

[84] L.G. Crespo, S.P. Kenny, and D.P. Giesy. Reliability analysis of polynomial systems subject to p-box uncertainties. *Mechanical Systems and Signal Processing*, 37(1-2):121–136, May 2013.

[85] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, Dec 1989.

[86] R.W.R. Darling. *Differential forms and connections*. Cambridge University Press, 1994.

[87] E.B. Davies and E.B. Davies. *Spectral theory and differential operators*, volume 42. Cambridge University Press, 1996.

[88] R.O. Davis and A.P.S. Selvadurai. *Plasticity and geomechanics*. Cambridge University Press, 2005.

[89] J.G. De Jalon and E. Bayo. *Kinematic and dynamic simulation of multibody systems: the real-time challenge*. Springer Science & Business Media, 2012.

[90] P. Del Moral and S. Penev. *Stochastic Processes: From Applications to Theory*. CRC Press, 2017.

[91] J.P. Delhomme. Spatial variability and uncertainty in groundwater flow parameters: a geostatistical approach. *Water Resources Research*, 15(2):269–280, 1979.

[92] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[93] G. Deodatis and R.C. Micaletti. Simulation of Highly Skewed Non-Gaussian Stochastic Processes. *J. Eng. Mech.*, 127(12):1284 – 1295, Dec 2001.

[94] A. Der Kiureghian. Analysis of structural reliability under parameter uncertainties. *Probabilistic Engineering Mechanics*, 23(4):351–358, Oct 2008.

[95] S.L. Derby and R.L. Keeney. Risk Analysis: Understanding "How Safe is Safe Enough?". *Risk Analysis*, 1(3):217 – 224, Sep 1981.

[96] C.S. Desai and H.J. Siriwardane. *Constitutive laws for engineering materials with emphasis on geologic materials*. Prentice-Hall, 1984.

[97] R. DeVore, G. Kerkyacharian, D. Picard, and V. Temlyakov. Approximation methods for supervised learning. *Foundations of Computational Mathematics*, 6(1):3–58, Dec 2005.

[98] I. Doghri and D.S. Chandrasekharaiah. Mechanics of deformable solids: Linear and nonlinear, analytical and computational aspects. *Applied Mechanics Reviews*, 54:B105, 2001.

[99] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer Science + Business Media, 2001.

[100] P.M. Doyen. Porosity from seismic data: A geostatistical approach. *Geophysics*, 53(10):1263–1275, 1988.

[101] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[102] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, 2015.

[103] H. El-Ramly, N.R. Morgenstern, and D.M. Cruden. Probabilistic slope stability analysis for practice. *Can. Geotech. J.*, 39(3):665 – 683, Jun 2002.

[104] M. Eldred and J. Burkardt. Comparison of Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Quantification. *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, Jan 2009.

[105] M.S. Eldred and J. Burkardt. Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification. *AIAA paper*, 976(2009):1–20, 2009.

[106] I. Elishakoff. *Probabilistic Methods in the Theory of Structures*. A Wiley-Interscience publication. Wiley, 1983.

[107] D. Elmo and D. Stead. An integrated numerical modelling–discrete fracture network approach applied to the characterisation of rock mass strength of naturally fractured pillars. *Rock Mechanics and Rock Engineering*, 43(1):3–19, 2010.

[108] S.S. Epp. *Discrete mathematics with applications*. Cengage Learning, 4 edition, 2010.

[109] B.K. Epperson and T. Li. Measurement of genetic structure within populations using moran's spatial autocorrelation statistics. *Proceedings of the National Academy of Sciences*, 93(19):10528–10532, 1996.

[110] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[111] J.D. Faires and R.L. Burden. *Numerical Methods*. Cengage Learning, 4 edition, 2012.

[112] H. Fang and D.P. O'Leary. Modified Cholesky algorithms: a catalog with new approaches. *Math. Program.*, 115(2):319–349, Jul 2007.

[113] I. Farmaga, P. Shmigelskyi, P. Spiewak, and L. Ciupinski. Evaluation of computational complexity of finite element analysis. In *CAD Systems in Microelectronics (CADSM), 2011 11th International Conference The Experience of Designing and Application of*, pages 213–214. IEEE, 2011.

[114] C. Feng, H. Wang, and X.M. Tu. Geometric mean of nonnegative random variable. *Communications in Statistics-Theory and Methods*, 42(15):2714–2717, 2013.

[115] G.A. Fenton. Error Evaluation of Three Random Field Generators. *J. Eng. Mech.*, 120(12):2478–2497, Dec 1994.

[116] G.A. Fenton and D.V. Griffiths. Bearing capacity prediction of spatially random $c - \phi$ soils. *Can. Geotech. J.*, 40(1):54–65, Feb 2003.

[117] G.A. Fenton and D.V. Griffiths. Review of Probability Theory, Random Variables, and Random Fields. *Probabilistic Methods in Geotechnical Engineering*, pages 1–69, 2007.

[118] G.A. Fenton and D.V. Griffiths. *Risk Assessment in Geotechnical Engineering*. John Wiley & Sons, Inc., Aug 2008.

[119] G.A. Fenton and E.H. Vanmarcke. Simulation of Random Fields via Local Average Subdivision. *J. Eng. Mech.*, 116(8):1733–1749, Aug 1990.

[120] S. Ferrari, K. Rudd, and G. Di Muro. A constrained backpropagation approach to function approximation and approximate dynamic programming. *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, pages 162–181, 2013.

[121] M.A.T. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1150–1159, Sep 2003.

[122] P. Flach. *Machine Learning*. Cambridge University Press (CUP), 1 edition, 2012.

[123] C.W. Fox and S.J. Roberts. A tutorial on variational Bayesian inference. *Artificial Intelligence Review*, 38(2):85–95, Jun 2011.

[124] M.P. Frank. The physical limits of computing. *Computing in Science & Engineering*, 4(3):16–26, 2002.

[125] T. Frankel. *The Geometry of Physics*. Cambridge University Press, 2011.

[126] J. Franklin. *Computational methods for physics*. Cambridge University Press, 2013.

[127] J. Franklin and A. Daoud. *Proof in Mathematics: An Introduction*. Kew Books, 2010.

[128] D. Freedman and P. Diaconis. On the histogram as a density estimator: L2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57(4):453–476, Dec 1981.

[129] M. Friendly, G. Monette, and J. Fox. Elliptical Insights: Understanding Statistical Methods through Elliptical Geometry. *Statistical Science*, 28(1):1–39, Feb 2013.

[130] J. Garcke. Sparse grids in a nutshell. *Sparse Grids and Applications*, pages 57–80, 2012.

[131] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandiè. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design*, 239(10):1768–1778, Oct 2009.

[132] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandiè. Mohr-coulomb plasticity. *MOOSE: A parallel computational framework for coupled systems of nonlinear equations - online documentation*, 2015.

[133] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandiè. Plasticity and the return-mapping algorithm. *MOOSE: A parallel computational framework for coupled systems of nonlinear equations - online documentation*, 2016.

[134] A.E. Gelfand. Gibbs Sampling. *Journal of the American Statistical Association*, 95(452):1300–1304, Dec 2000.

[135] L.W. Gelhar and C.L. Axness. Three dimensional stochastic analysis of macrodispersion in aquifers. *Water Resources Research*, 19(1):161–180, 1983.

[136] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin. *Bayesian data analysis*, volume 2. CRC press Boca Raton, FL, 2014.

[137] J.E. Gentle, W.K. Härdle, and Y. Mori. *Handbook of Computational Statistics: Concepts and Methods*. Springer Handbooks of Computational Statistics. Springer Berlin Heidelberg, 2012.

[138] A. Genz and F. Bretz. *Computation of Multivariate Normal and t Probabilities*. Springer Berlin Heidelberg, 2009.

[139] T. Gerstner and M. Griebel. Dimension-adaptive tensor-product quadrature. *Computing*, 71(1):65–87, aug 2003.

[140] C.J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):473–483, nov 1992.

[141] R.G. Ghanem and P.D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer New York, 1991.

[142] R.G. Ghanem and P.D. Spanos. Spectral techniques for stochastic finite elements. *ARCO*, 4(1):63–100, Mar 1997.

[143] G.P. Giani. *Rock Slope Stability Analysis*. Taylor & Francis, 1992.

[144] W.R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC Interdisciplinary Statistics. Taylor & Francis, 1995.

[145] D.T. Gillespie. Exact numerical simulation of the ornstein-uhlenbeck process and its integral. *Phys. Rev. E*, 54(2):2084–2091, Aug 1996.

[146] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[147] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[148] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 4 edition, 2013.

[149] O. Gonzalez and A.M. Stuart. *A First Course in Continuum Mechanics*. Cambridge University Press, 2001.

[150] O. Gonzalez and A.M. Stuart. *A First Course in Continuum Mechanics*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2008.

[151] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning Series. MIT Press, 2016.

[152] L.A. Goodman. On the exact variance of products. *Journal of the American Statistical Association*, 55(292):708–713, dec 1960.

[153] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[154] A. Graves, A. Mohamed, and G.E. Hinton. Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013.

[155] M. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica, Second Edition*. Textbooks in Mathematics. Taylor & Francis, 1997.

[156] D.K.E. Green. Markov chain monte carlo for rare event reliability analysis with nonlinear finite elements. In *SIAM Conference on Uncertainty Quantification*, Lausanne, Switzerland, 2016.

[157] D.K.E. Green. Efficient Markov chain Monte Carlo for combined subset simulation and nonlinear finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 313:337–361, Jan 2017.

[158] D.K.E. Green, K. Douglas, and G. Mostyn. The simulation and discretisation of random fields for probabilistic finite element analysis of soils using meshes of arbitrary triangular elements. *Computers and Geotechnics*, 68:91–108, Jul 2015.

[159] D.V. Griffiths and G.A. Fenton. Probabilistic Settlement Analysis by Stochastic and Random Finite-Element Methods. *Journal of Geotechnical and Geoenvironmental Engineering*, 135(11):1629–1637, Nov 2009.

[160] M. Grigoriu. Simulation of stationary non-Gaussian translation processes. *Journal of Engineering Mechanics*, 124(2):121–sch126, Feb 1998.

[161] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, 2001.

[162] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, and S. Zaleski. Volume-of-fluid interface tracking with smoothed surface stress methods for three dimensional flows. *Journal of Computational physics*, 152(2):423–456, 1999.

[163] V. Guillemin and A. Pollack. *Differential topology*, volume 370. American Mathematical Soc., 2010.

[164] J. Hadamard. *Le problème de Cauchy et les équations aux dérivées partielles linéaires hyperboliques: leçons professées à l'Université Yale*. Hermann et cie, 1932.

[165] V. Hakim and A. Karma. Laws of crack motion and phase-field models of fracture. *Journal of the Mechanics and Physics of Solids*, 57(2):342–368, 2009.

[166] A. Hald. On the history of maximum likelihood in relation to inverse probability and least squares. *Statistical Science*, 14(2):214–222, May 1999.

[167] L.J. Halliwell. The lognormal random multivariate. In *Casualty Actuarial Society E-Forum; Spring*, 2015.

[168] P.R. Halmos. *Measure Theory*. Graduate Texts in Mathematics. Springer New York, 2013.

[169] J.H. Halton. Sequential Monte Carlo. *Mathematical Proceedings of the Cambridge Philosophical Society*, 58(01):57, Jan 1962.

[170] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[171] M.E. Harr. *Reliability-based Design in Civil Engineering*. Dover books on mathematics. Dover Publications, 1996.

[172] A. Hatcher. *Algebraic Topology*. Algebraic Topology. Cambridge University Press, 2002.

[173] S. Haykin. *Neural Networks: A comprehensive foundation*. Prentice Hall, 2 edition, 2004.

[174] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[175] P. Hennig, M.A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. In *Proc. R. Soc. A*, volume 471. The Royal Society, 2015.

[176] N. Henze. A probabilistic representation of the'skew-normal'distribution. *Scandinavian journal of statistics*, pages 271–275, 1986.

[177] M. Herzog, A. Gilg, M. Paffrath, P. Rentrop, and U. Wever. Intrusive versus non-intrusive methods for stochastic finite elements. *From Nano to Space*, pages 161–174, 2008.

[178] N.J. Higham. Accuracy and Stability of Numerical Algorithms. Jan 2002.

[179] A. Hinrichs, E. Novak, M. Ullrich, and H. Woźniakowski. The curse of dimensionality for numerical integration of smooth functions. *Mathematics of Computation*, 83(290):2853–2863, 2014.

[180] G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, Aug 2002.

[181] G.E. Hinton. A practical guide to training restricted boltzmann machines. *Neural Networks: Tricks of the Trade*, pages 599–619, 2012.

[182] G.E. Hinton, P. Dayan, B. Frey, and R.M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, May 1995.

[183] G.E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, Jul 2006.

[184] G.E. Hinton and R.R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, Jul 2006.

[185] Y.-C. Ho and D.L. Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570, 2002.

[186] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997.

[187] E. Hoek. *Practical rock engineering*. RocScience, 2000.

[188] T. Hofmann, B. Schölkopf, and A. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, Jun 2008.

[189] H. Holden, B. Øksendal, J. Ubøe, and T. Zhang. Stochastic partial differential equations. In *Stochastic partial differential equations*, pages 141–191. Springer, 1996.

[190] Y. Honjo and K.-K. Phoon. *Monte Carlo simulation in reliability analysis*, chapter 4, pages 169–191. CRC Press, 2008.

[191] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Matrix Analysis. Cambridge University Press, 2012.

[192] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, Jan 1991.

[193] J. Huang, G.A. Fenton, D.V. Griffiths, D. Li, and C. Zhou. On the efficient estimation of small failure probability in slopes. *Landslides*, 14(2):491–498, 2017.

[194] M. Huber, P. Vermeer, and A. Bárdossy. Evaluation of soil variability and its consequences. *Numerical Methods in Geotechnical Engineering*, pages 363–368, Jun 2000.

[195] J.K. Hunter. Lecture notes on applied mathematics: Methods and models. Math 280 Lecture Notes: Department of Mathematics, University of California, Davis, 2009.

[196] IEEE. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008.

[197] R.L. Iman. *Latin hypercube sampling*. Wiley Online Library, 2008.

[198] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press (CUP), 2008.

[199] H. Jaeger, W. Maass, and J. Principe. Special issue on echo state networks and liquid state machines. *Neural Networks*, 20(3):287–289, Apr 2007.

[200] M. Jamil and X.S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150, 2013.

[201] R. Jankowski. Non-linear fem analysis of pounding-involved response of buildings under non-uniform earthquake excitation. *Engineering Structures*, 37:99–105, 2012.

[202] W. Jaunzemis. *Continuum mechanics*. Macmillan series in applied mechanics. Macmillan, 1967.

[203] B. Jeremić, K. Sett, and M.L. Kavvas. Probabilistic elasto-plasticity: formulation in 1D. *Acta Geotechnica*, 2(3):197–210, oct 2007.

[204] E. Jones, T.E. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2017-03-08].

[205] A.G. Journel. Geostatistics for conditional simulation of ore bodies. *Economic Geology*, 69(5):673–687, 1974.

[206] A. Kabán. On Bayesian classification with Laplace priors. *Pattern Recognition Letters*, 28(10):1271–1282, Jul 2007.

[207] A. Karma, D.A. Kessler, and H. Levine. Phase-field model of mode iii dynamic fracture. *Physical Review Letters*, 87(4):045501, 2001.

[208] G. Karniadakis and S. Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.

[209] K.S. Kasiviswanathan, K.P. Sudheer, and J. He. Quantification of prediction uncertainty in artificial neural network models. *Studies in Computational Intelligence*, pages 145–159, 2016.

[210] A. Keese and H.G. Matthies. Hierarchical parallelisation for the solution of stochastic finite element equations. *Computers & Structures*, 83(14):1033–1047, May 2005.

[211] A. Khosravi, S. Nahavandi, and D. Creighton. Quantifying uncertainties of neural network-based electricity price forecasts. *Applied Energy*, 112:120–129, Dec 2013.

[212] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

[213] P.E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2011.

[214] D.E. Knuth. The art of computer programming, 3rd edn. seminumerical algorithms, vol. 2, 1997.

[215] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press, 2009.

[216] A.N. Kolmogorov. Foundations of the theory of probability.

[217] A.N. Kolmogorov and S.V. Fomin. *Introductory Real Analysis (Dover Books on Mathematics)*. Dover Publications, 1975.

[218] A. Kolobov. Planning with markov decision processes: An ai perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, 2012.

[219] S. Krantz and H. Parks. *Geometric Integration Theory*. Springer Nature, 2008.

[220] D. Krewski, M. Jerrett, R.T. Burnett, R. Ma, E. Hughes, Y. Shi, M.C. Turner, C. Arden Pope III, G. Thurston, E.E. Calle, et al. *Extended follow-up and spatial analysis of the American Cancer Society study linking particulate air pollution and mortality*. Number 140. Health Effects Institute Boston, MA, 2009.

[221] E. Kreyszig. *Introductory Functional Analysis with Applications*. Wiley, 1989.

[222] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, 2010.

[223] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[224] D.P. Kroese and J.C.C. Chan. *Statistical Modeling and Computation*. Springer New York, 2014.

[225] S. Kullback. *Information Theory and Statistics*. Dover books on intermediate and advanced mathematics. Peter Smith, 1968.

[226] K. Kunen. *Set theory an introduction to independence proofs*, volume 102. Elsevier, 2014.

[227] F.Y. Kuo, C. Schwab, and I.H. Sloan. Quasi-Monte Carlo Finite Element Methods for a Class of Elliptic Partial Differential Equations with Random Coefficients. *SIAM Journal on Numerical Analysis*, 50(6):3351 – 3374, Jan 2012.

[228] L.D. Landau and E.M. Lifshitz. *Statistical Physics*. Number v. 5. Elsevier Science, 2013.

[229] C. Lantuéjoul. *Geostatistical simulation: models and algorithms*. Springer Science & Business Media, 2013.

[230] D. Laurent, O. Legrand, P. Sebbah, C. Vanneste, and F. Mortessagne. Localized modes in a finite-size open disordered microwave cavity. *Physical review letters*, 99(25):253902, 2007.

[231] O. Le Maître and O.M. Knio. *Spectral methods for uncertainty quantification: with applications to computational fluid dynamics*. Springer Science & Business Media, 2010.

[232] Y. LeCun, Y. Bengio, and G.E. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[233] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F.-J. Huang. *Energy-based Models*, chapter 10. MIT Press, 2007.

[234] Y.A. LeCun, L. Bottou, G.B. Orr, and K.-R. M uller. Efficient backprop (republished from 1998 version). *Neural Networks: Tricks of the Trade*, pages 9–48, 2012.

[235] H. Lee, A. Battle, R. Raina, and A.Y. Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2007.

[236] J.M. Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–29. Springer, 2003.

[237] Y.-M. Lee and J.H. Ellis. Estimation and simulation of lognormal random fields. *Computers & Geosciences*, 23(1):19–31, 1997.

[238] D. Li, T. Xiao, Z. Cao, and C. Zhou. Subset simulation-based random finite element method for slope reliability analysis and risk assessment. In T. Haukaas, editor, *Proceedings of the 12th International Conference on Applications of Statistics and Probability in Civil Engineering (ICASP12)*, Vancouver, Canada, July 2015.

[239] D.-Q. Li, T. Xiao, Z.-J. Cao, C.-B. Zhou, and L.-M. Zhang. Enhancement of random finite element method in reliability analysis and risk assessment of soil slopes using Subset Simulation. *Landslides*, Mar 2015.

[240] H. Li and D. Zhang. Probabilistic collocation method for flow in porous media: Comparisons with other stochastic methods. *Water Resources Research*, 43(9), Sep 2007.

[241] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Nature, 1997.

[242] M. Li, T. Zhang, Y. Chen, and A. Smola. Efficient mini-batch training for stochastic optimization. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 2014*, 2014.

[243] A. Lisjak and G. Grasselli. A review of discrete modeling techniques for fracturing processes in discontinuous rock masses. *Journal of Rock Mechanics and Geotechnical Engineering*, 6(4):301–314, 2014.

[244] A. Litvinenko, H.G. Matthies, and T.A. El-Moselhy. Sampling and Low-Rank Tensor Approximation of the Response Surface. *Springer Proceedings in Mathematics & Statistics*, pages 535–551, 2013.

[245] S. Lloyd. Ultimate physical limits to computation. *arXiv preprint quant-ph/9908043*, 1999.

[246] A. Logg, K.-A. Mardal, and G. Wells. Automated solution of differential equations by the finite element method. *Lecture Notes in Computational Science and Engineering*, 84, 2012.

[247] S. Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

[248] L.E. Malvern. *Introduction to the mechanics of a continuous medium*. Prentice-Hall series in engineering of the physical sciences. Prentice-Hall, 1969.

[249] E. Marks. A Note on a Geometric Interpretation of the Correlation Coefficient. *Journal of Educational Statistics*, 7(3):233, 1982.

[250] J.E. Marsden and T.J.R. Hughes. *Mathematical Foundations of Elasticity*. Dover Civil and Mechanical Engineering. Dover Publications, 2012.

[251] J.E. Marsden and A. Tromba. *Vector calculus*. Macmillan, 2003.

[252] G. Matheron. *Estimating and Choosing - An Essay on Probability in Practice*. Springer Berlin Heidelberg, 1989.

[253] H.G. Matthies. Uncertainty Quantification with Stochastic Finite Elements. *Encyclopedia of Computational Mechanics*, Nov 2004.

[254] H.G. Matthies. Stochastic finite elements: Computational approaches to stochastic partial differential equations. *ZAMM*, 88(11):849–873, Nov 2008.

[255] H.G. Matthies, C.E. Brenner, C.G. Bucher, and C.G. Soares. Uncertainties in probabilistic numerical analysis of structures and solids-stochastic finite elements. *Structural safety*, 19(3):283–336, 1997.

[256] H.G. Matthies and A. Keese. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(12-16):1295–1331, Apr 2005.

[257] H.G. Matthies, E. Zander, B.V. Rosić, and A. Litvinenko. Parameter estimation via conditional expectation: a Bayesian inversion. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1), Aug 2016.

[258] J.R. McDonnell and D. Waagen. Evolving neural network architecture. Technical report, Naval command Control and Ocean Surveillance Center RDT AND E DIV San Diego CA, 1993.

[259] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[260] J.D. Meiss. *Differential Dynamical Systems*. Society for Industrial and Applied Mathematics, Jan 2007.

[261] E. Merzbach and D. Nualart. A characterization of the spatial poisson process and changing time. *The Annals of Probability*, pages 1380–1390, 1986.

[262] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087, 1953.

[263] B.R. Meyer, L.W. Bazan, et al. A discrete fracture network model for hydraulically induced fractures-theory, parametric and case studies. In

SPE Hydraulic Fracturing Technology Conference. Society of Petroleum Engineers, 2011.

[264] C. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Jan 2000.

[265] M.A. Meyers and K.K. Chawla. *Mechanical Behavior of Materials*. Cambridge University Press, 2 edition, 2009.

[266] S.P. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability*. Springer Science + Business Media, 1993.

[267] L. Mihalkova, T. Huynh, and R.J. Mooney. Mapping and revising markov logic networks for transfer learning. In *AAAI*, volume 7, pages 608–614, 2007.

[268] T. Minka. Bayesian linear regression. Technical report, Technical report, MIT, 2000.

[269] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

[270] G.R. Mostyn and K.S. Li. Probabilistic slope analysis – state of play. In *Proceedings of the conference on probabilistic methods in geotechnical engineering.*, pages 89–109, Rotterdam: Balkema, 1993.

[271] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press (CUP), 1995.

[272] D. Muir Wood. *Soil Behaviour and Critical State Soil Mechanics*. Cambridge University Press (CUP), 1991.

[273] J.R. Munkres. *Elements of algebraic topology*, volume 2. Addison-Wesley Menlo Park, 1984.

[274] J.R. Munkres. *Topology*. Prentice Hall, 2000.

[275] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive computation and machine learning series. MIT Press, 2012.

[276] V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[277] M. Nakahara. *Geometry, Topology and Physics, Second Edition*. Taylor & Francis, Jun 2003.

[278] R.M. Neal and G.E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[279] R.D. Neidinger. Introduction to automatic differentiation and matlab object-oriented programming. *SIAM Review*, 52(3):545–563, Jan 2010.

[280] R.B. Nelsen. *An introduction to copulas*. Springer Science & Business Media, 2007.

[281] A. Neubauer. Tikhonov regularisation for non-linear ill-posed problems: optimal convergence rates and finite-dimensional approximation. *Inverse Problems*, 5(4):541–557, Aug 1989.

[282] A.Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. *Twenty-first international conference on Machine learning - ICML '04*, 2004.

[283] W.F. Noh and P. Woodward. Slic (simple line interface calculation). In *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28–July 2, 1976 Twente University, Enschede*, pages 330–340. Springer, 1976.

[284] W. Ochs. Basic properties of the generalized boltzmann-gibbs- Shannon entropy. *Reports on Mathematical Physics*, 9(2):135–155, Apr 1976.

[285] J.T. Oden and L. Demkowicz. *Applied Functional Analysis, Second Edition*. Chapman and Hall/CRC, 2010.

[286] A. O'Hagan and T. Leonard. Bayes estimation subject to uncertainty about parameter constraints. *Biometrika*, 63(1):201–203, 1976.

[287] T.E. Oliphant. *Python for Scientific Computing*, volume 9. Institute of Electrical and Electronics Engineers (IEEE), 2007.

[288] B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[289] M. Opper and D. Saad. *Advanced mean field methods: Theory and practice*. MIT press, 2001.

[290] I. Papaioannou, W. Betz, K. Zwirglmaier, and D. Straub. MCMC algorithms for Subset Simulation. *Probabilistic Engineering Mechanics*, 41:89–103, Jul 2015.

[291] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[292] J. Pearl. Graphical models for probabilistic and causal reasoning. In *Quantified representation of uncertainty and imprecision*, pages 367–389. Springer, 1998.

[293] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[294] R. Penrose. *The emperor's new mind: Concerning computers, minds, and the laws of physics*. Oxford Paperbacks, 1999.

[295] K.-K. Phoon, editor. *Reliability-Based Design in Geotechnical Engineering: Computations and Applications*. Taylor & Francis, 2008.

[296] K.-K. Phoon, H.W. Huang, and S.T. Quek. Simulation of strongly non-Gaussian processes using karhunen–loève expansion. *Probabilistic Engineering Mechanics*, 20(2):188–198, Apr 2005.

[297] K.-K. Phoon and F.H. Kulhawy. Characterization of geotechnical variability. *Can. Geotech. J.*, 36(4):612–624, Nov 1999.

[298] J. Pieczyńska, W. Puła, D.V. Griffiths, and G.A. Fenton. Probabilistic characteristics of strip footing bearing capacity evaluated by random finite element method. *Applications of Statistics and Probability in Civil Engineering - Faber, Köhler & Nishijima (eds.)*, pages 1673–1682, Jul 2011.

[299] S. Pietruszczak. *Fundamentals of plasticity in geomechanics*. Crc Press, 2010.

[300] F.J. Pinski, G. Simpson, A.M. Stuart, and H. Weber. Algorithms for Kullback–Leibler approximation of probability measures in infinite dimensions. *SIAM Journal on Scientific Computing*, 37(6):A2733–A2757, Jan 2015.

[301] M. Pitt, D. Chan, and R. Kohn. Efficient bayesian inference for gaussian copula regression models. *Biometrika*, 93(3):537–554, 2006.

[302] D. Pollard. *A user's guide to measure theoretic probability*, volume 8. Cambridge University Press, 2002.

[303] J. Poulson, B. Marker, R.A. van de Geijn, J.R. Hammond, and N.A. Romero. Elemental. *TOMS*, 39(2):1–24, Feb 2013.

[304] W.B. Powell. *Approximate Dynamic Programming*. John Wiley & Sons, Inc., Aug 2011.

[305] W.H. Press, S. A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[306] F. Radjaï and F. Dubois. *Discrete-element modeling of granular materials*. Wiley-Iste, 2011.

[307] L.B. Rall. *Automatic Differentiation: Techniques and Applications*. Springer Berlin Heidelberg, 1981.

[308] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[309] M. Reed and B. Simon. *Methods of Modern Mathematical Physics I: Functional Analysis. Revised and enlarged edition*. Academic Press, 1980.

[310] M. Renardy and R.C. Rogers. *An introduction to partial differential equations*, volume 13. Springer Science & Business Media, 2006.

[311] D. Revuz. *Markov Chains*. North-Holland Mathematical Library. Elsevier Science, 2008.

[312] M.E. Ricotti and E. Zio. Neural network approach to sensitivity and uncertainty analysis. *Reliability Engineering & System Safety*, 64(1):59–71, Apr 1999.

[313] C. Robert and G. Casella. A Short History of MCMC. *Handbook of Markov Chain Monte Carlo*, May 2011.

[314] C.P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer New York, 1999.

[315] V.K. Rohatgi and A.K.MD. Ehsanes Saleh. *An Introduction to Probability and Statistics*. John Wiley & Sons, Inc., sep 2000.

[316] S.M. Ross. *Stochastic processes*. Wiley, New York, 1996.

[317] G.G. Roussas. *An introduction to measure-theoretic probability*. Academic Press, 2014.

[318] C. Roy and W. Oberkampf. A complete framework for verification, validation, and uncertainty quantification in scientific computing (invited). *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Jan 2010.

[319] R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method*. Springer New York, 2004.

[320] R.Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Statistics. Wiley, 2011.

[321] K. Rudd and S. Ferrari. A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 155:277–285, 2015.

[322] L. Rüschendorf. *Mathematical risk analysis*. Springer Series in Operations Research and Financial Engineering. Springer, 2013.

[323] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Always learning. Pearson Education, Limited, 2014.

[324] S. Sakamoto and R. Ghanem. Simulation of multi-dimensional non-gaussian non-stationary random fields. *Probabilistic Engineering Mechanics*, 17(2):167–176, Apr 2002.

[325] C.A. Schenk and G.I. Schuëller. Uncertainty Assessment of Large Finite Element Systems. *Lecture Notes in Applied and Computational Mechanics*, 2005.

[326] J. Schmid. The Relationship between the Coefficient of Correlation and the Angle included between Regression Lines. *The Journal of Educational Research*, 41(4):311–313, Dec 1947.

[327] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, Jan 2015.

[328] R. Schobi, B. Sudret, and J. Wiart. Polynomial-chaos-based kriging. *International Journal for Uncertainty Quantification*, 5(2):171–193, 2015.

[329] G.I. Schuëller and H.J. Pradlwarter. Benchmark study on reliability estimation in higher dimensions of structural systems – an overview. *Structural Safety*, 29(3):167–182, jul 2007.

[330] D.W. Scott. *Multivariate Density Estimation*. John Wiley & Sons, Inc, Mar 2015.

[331] A.P.S. Selvadurai. *Partial Differential Equations in Mechanics 1: Fundamentals, Laplace's Equation, Diffusion Equation, Wave Equation*. Number v. 1. Springer Berlin Heidelberg, 2013.

[332] A.P.S. Selvadurai. *Partial Differential Equations in Mechanics 2: The Biharmonic Equation, Poisson's Equation*. Number v. 2. Springer Berlin Heidelberg, 2013.

[333] P.A. Selvadurai and A.P.S. Selvadurai. On the effective permeability of a heterogeneous porous medium: the role of the geometric mean. *Philosophical Magazine*, 94(20):2318–2338, 2014.

[334] K. Sett, B. Jeremić, and M.L. Kavvas. Probabilistic elasto-plasticity: solution and verification in 1D. *Acta Geotechnica*, 2(3):211–220, oct 2007.

[335] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, Jul 1948.

[336] A. Shapiro and T. Homem-de Mello. On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs. *SIAM Journal on Optimization*, 11(1):70–86, Jan 2000.

[337] M. Shinozuka. Simulation of multivariate and multidimensional random processes. *The Journal of the Acoustical Society of America*, 49(1B):357–368, 1971.

[338] M. Shinozuka and G. Deodatis. Simulation of stochastic processes by spectral representation. *Applied Mechanics Reviews*, 44(4):191–204, 1991.

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

[340] J.C. Simo and T.J.R. Hughes. *Computational inelasticity*, volume 7. Springer Science & Business Media, 2006.

[341] M. Sklar. *Fonctions de Répartition À N Dimensions Et Leurs Marges*. Université Paris 8, 1959.

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

[343] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, Aug 2004.

[344] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M.M.A. Patwary, M. Prabhat, and R.P. Adams. Scalable bayesian optimization using deep neural networks. In *ICML*, pages 2171–2180, 2015.

[345] J. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*, volume 97 of *Applied Optimization*. Springer-Verlag, 2005.

[346] R. Solomonoff. Two kinds of probabilistic induction. *The Computer Journal*, 42(4):256–259, Apr 1999.

[347] J.C. Spall. *Introduction to Stochastic Search and Optimization*. Wiley-Blackwell, Mar 2003.

[348] R. Spatschek, E. Brener, and A. Karma. Phase field modeling of crack propagation. *Philosophical Magazine*, 91(1):75–95, 2011.

[349] A.J.M. Spencer. *Continuum Mechanics*. Dover books on physics. Dover Publications, 2004.

[350] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[351] G. Stefanou. The stochastic finite element method: Past, present and future. *Computer Methods in Applied Mechanics and Engineering*, 198(9-12):1031 – 1051, 2009.

[352] A. Steiger, R. Muller, A.R. Oliva, Y. Deng, Q. Sun, M. White, and J. Lehman. Terahertz laser power measurement comparison. *IEEE Transactions on Terahertz Science and Technology*, pages 1–6, 2016.

[353] G. Strang and G. Fix. *An Analysis of the Finite Element Method*. Prentice-Hall series in automatic computation. Wellesley-Cambridge Press, 2008.

[354] M. Strevens. *Macmillan Encyclopedia of Philosophy*, chapter The Bayesian Approach to the Philosophy of Science. Macmillan Reference USA, 2 edition, 2006.

[355] A.M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, May 2010.

[356] B. Sudret. *Uncertainty propagation and sensitivity analysis in mechanical models – Contributions to structural reliability and stochastic spectral methods*, 2007. Habilitation à diriger des recherches, Université Blaise Pascal, Clermont-Ferrand, France.

[357] B. Sudret. Meta-models for Structural Reliability and Uncertainty Quantification. *Proceedings of the 5th Asian-Pacific Symposium on Structural Reliability and its Applications*, 2012.

[358] B. Sudret, G. Blatman, and M. Berveiller. Response surfaces based on polynomial chaos expansions. *Construction Reliability*, pages 147–167, Feb 2013.

[359] B. Sudret and A. Der Kiureghian. *Stochastic finite element methods and reliability: a state-of-the-art report*. Department of Civil and Environmental Engineering, University of California, 2000.

[360] B. Sudret and A. Der Kiureghian. Comparison of finite element reliability methods. *Probabilistic Engineering Mechanics*, 17(4):337–348, Oct 2002.

[361] N.Z. Sun. *Inverse problems in groundwater modeling*, volume 6. Springer Science & Business Media, 2013.

[362] N.Z. Sun and A. Sun. *Mathematical Modeling of Groundwater Pollution*. Springer New York, 2013.

[363] P. Suppes. *Axiomatic set theory*. Courier Corporation, 1960.

[364] I. Sutskever, J. Martens, and G.E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[365] I. Sutskever, O. Vinyals, and Q.V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[366] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. A Bradford book. Bradford Book, 1998.

[367] A.P. Suvorov and A.P.S. Selvadurai. On poro-hyperelastic shear. *Journal of the Mechanics and Physics of Solids*, 96:445–459, 2016.

[368] S. Suzumura and R. Nakano. Complex-valued multilayer perceptron search utilizing eigen vector descent and reducibility mapping. *Lecture Notes in Computer Science*, pages 1–8, 2012.

[369] S. Tadelis. *Game theory: an introduction*. Princeton University Press, 2013.

[370] U. Tautenhahn and Q. Jin. Tikhonov regularization and a posteriori rules for solving nonlinear ill posed problems. *Inverse Problems*, 19(1):1–21, Nov 2002.

[371] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[372] A.N. Tikhonov and V.I.A. Arsenin. *Solutions of ill-posed problems*. Scripta series in mathematics. Winston, 1977.

[373] A.N. Tikhonov, A.S. Leonov, and A.G. Yagola. Nonlinear ill-posed problems. 1998.

[374] M.E. Tipping and C.M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 21/3:611–622, January 1999.

[375] Y.L. Tong. *The Multivariate Normal Distribution*. Springer New York, 1990.

[376] L. Torrey and J. Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends*, pages 242–264, 2009.

[377] J.A. Trangenstein. *Numerical Solution of Elliptic and Parabolic Partial Differential Equations*. Cambridge University Press, 2013.

[378] G.E. Uhlenbeck and L.S. Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

[379] D. Van Dalen. *Logic and structure*. Springer, 2004.

[380] S. van der Walt, S.C. Colbert, and G. Varoquaux. The NumPy Array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, Mar 2011.

[381] E.H. Vanmarcke. *Random Fields: Analysis and Synthesis*. World Scientific, 2 edition, Jul 2010.

[382] E.H. Vanmarcke, E. Heredia-Zavoni, and G.A. Fenton. Conditional simulation of spatially correlated earthquake ground motion. *Journal of Engineering Mechanics*, 119(11):2333–2352, 1993.

[383] V.N. Vapnik. *Statistical learning theory*. Adaptive and learning systems for signal processing, communications, and control. Wiley, 1998.

[384] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Nature, 2000.

[385] A.N. Vasil'ev. *The field theoretic renormalization group in critical behavior theory and stochastic dynamics*. CRC press, 2004.

[386] H.K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[387] G. Vessia, C. Cherubini, J. Pieczyńska, W. Puła, et al. Application of random finite element method to bearing capacity design of strip footing. *Journal of GeoEngineering*, 4(3):103–112, 2009.

[388] J.-P. Vila, V. Wagner, and P. Neveu. Bayesian nonlinear model selection and neural networks: a conjugate prior approach. *IEEE Transactions on neural networks*, 11(2):265–278, 2000.

[389] T. Von Kármán and M.A. Biot. *Mathematical methods in engineering: an introduction to the mathematical treatment of engineering problems.* McGraw-Hill book company, inc., 1940.

[390] M. Vořechovskỳ and D. Novák. Simulation of random fields for stochastic finite element analyses. In *Proc. 9th International Conference on Structural Safety and Reliability (ICOSSAR 2005)*, volume 5, pages 2545 – 2552, 2005.

[391] D.J. Wales and T.V. Bogdan. Potential energy and free energy landscapes. *The Journal of Physical Chemistry B*, 110(42):20765–20776, Oct 2006.

[392] P.J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, Jan 1988.

[393] N. Wiener. *Nonlinear Problems in Random Theory.* Technology Press research monographs. Technology Press of Massachusetts Institute of Technology, 1958.

[394] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications.* Extrapolation, Interpolation, and Smoothing of Stationary Time Series. M.I.T. Press, 1964.

[395] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement Learning*, pages 5–32, 1992.

[396] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Trans. Evol. Computat.*, 1(1):67–82, Apr 1997.

[397] D.H. Wolpert and W.G. Macready. Coevolutionary Free Lunches. *IEEE Trans. Evol. Computat.*, 9(6):721–735, Dec 2005.

[398] D.H. Wolpert, W.G. Macready, et al. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[399] D. Xiu and G.E. Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, Jan 2002.

[400] M.D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science*, pages 818–833, 2014.

[401] W.Q. Zhu and G.Q. Cai. Generation of non-Gaussian stochastic processes using nonlinear filters. *Probabilistic Engineering Mechanics*, 36:56–62, Apr 2014.

[402] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *Finite Element Method: Its Basis and Fundamentals, The: Its Basis and Fundamentals*. Elsevier, Incorporated, 2013.

[403] D. Zwillinger and S. Kokoska. *CRC Standard Probability and Statistics Tables and Formulae*. Informa UK Limited, Dec 1999.