Uncertainty Quantification and Management in Multidisciplinary Design Optimisation

Joseph Loxham

A thesis presented for the degree of

Doctor of Philosophy

School of Aerospace, Transport and Manufacturing

Cranfield University

United Kingdom

Abstract

We analyse the uncertainty present at the structural-sizing stage of aircraft design due to interactions between aeroelastic loading and incomplete structural definition. In particular, we look at critical load case identification: the process of identifying the flight conditions at which the maximum loading conditions occur from sparse, expensive to obtain data. To address this challenge, we investigate the construction of *robust emulators*: probabilistic models of computer code outputs, which explicitly and reliably model their predictive uncertainty. Using Gaussian process regression, we show how such models can be derived from simple and intuitive considerations about the interactions between parameter inference and data, and via state-of-the-art statistical software, develop a generally applicable and easy to use method for constructing them. The effectiveness of these models is demonstrated on a range of synthetic and engineering test functions. We then use them to approach two facets of critical load case identification: sample efficient searching for the critical cases via Bayesian optimisation, and probabilistic assessment of *possible* locations for the critical cases from a given sample; the latter facilitating quantitative downselection of candidate load cases by ruling out regions of the search space with a low probability of containing the critical cases, potentially saving a designer many hours of simulation time. Finally, we show how the presence of design variability in the loads analysis implies a stochastic process, and attempt to construct a model for this by parametrisation of its marginal distributions.

Acknowledgements

This work is funded by a joint grant from EPSRC and Airbus UK.

First and foremost, I would like to thank my two academic supervisors; Dr Kipouros and Professor Savill, for both their guidance and their patience. I have been fortunate enough to have had both the freedom to explore tangents *and* have close supervision, when requested. This has contributed significantly to the extent to which I have enjoyed conducting this research: I hope it shows.

I would also like to offer greatly extended thanks to my two industrial supervisors and friends, Simon Coggon and Sanjiv Sharma. I have enjoyed every moment of my time on site at Airbus Filton, and this work would not exist without the continued input from both of you: Simon, my requests for feedback and advice have been *relentless* and I am very appreciative of all the time you have managed to squeeze out of your already manic schedule for me. Sanjiv, I have very much enjoyed your infectious enthusiasm for exploring unusual (and at times *esoteric*) solutions to tricky problems. I have learnt a lot from both of you.

I must also thank Gillian Hargreaves at Cranfield for all of the administrative support I have received over the course of my studies, particularly those requests for assistance made at the last-minute!

Finally, there are also many other people at Airbus Filton whose inputs have made important contributions to one or more aspects of this project. In no particular order, I must also thank Murray Cross, Ben Earl, Clive Staines, John Welch, Lucian Iorga, Vincent Malmedy, and Bennet Leong, along with the entire EGDCU Filton Team.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations:

BO	Bayesian Optimisation
CG	Centre of Gravity (as a fraction of the aerodynamic chord)
CV	Cross Validation
DOF	Degree(s) Of Freedom
EI	Expected Improvement
ELPPD	Expected Log Pointwise Predictive Density
EQ	Exponentiated Quadratic (covariance kernel)
GP	Gaussian Process
HMC	Hamiltonian Monte-Carlo
HTP	Horizontal Tail Plane
IQ	Interesting Quantity
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno (optimisation algorithm)
LHS	Latin-Hypercube Sample
LOO	Leave-One-Out
LPD	Log Predictive Density
LPPD	Log Pointwise Predictive Density
MAN	Manoeuvre (as relating to a load case)
MAP	Maximum A-posteriori Probability
MCMC	Markov-Chain Monte-Carlo
MES	Max-value Entropy Search
MLE	Maximum Likelihood Estimate

NUTS	No-U-Turn Sampler
PCA	Principal Component Analysis
PDF	Probability Density Function
PES	Predictive Entropy Search
PX	Pure Exploration
RMSE	Root Mean Square Error
SLSQP	Sequential Linear Least Squares Programming
SVD	Singular Value Decomposition
VI	Variational Inference

Symbols and Expressions:

α	Marginal standard deviation
Δ_i	(Minimum) distance between points in dimension i.
$\Gamma(\cdot)$	Regular gamma function
$\Gamma(\cdot, \cdot)$	Incomplete gamma function
$\hat{\mathbf{x}}$	Point with highest utility
$\hat{l}(\cdot)$	Surrogate loads process
$\hat{y}^*$	Predicted maximum
$\mathbf{P}$	PCA matrix
$\mathbf{Q}$	$N \times R$ array representing a low rank approximation of data matrix $\mathbf{Y}$
$\mathbf{q}$	Vector corresponding to a single column of the low rank approximation, $\mathbf{Q}$
$\mathbf{t}$	Vector of internal forces (stresses)
$\mathbf{X}$	$N \times D$ array; N points in D dimensional space
$\mathbf{x}$	Vector representing a point in D dimensional space (abstract) or a parametrised load case (loads)
$\mathbf{x}'$	Vector representing a second point in D dimensional space
$\mathbf{x}^*$	Critical load case
$\mathbf{y}$	Vector of responses (abstract) or forces (loads)
$\mathbf{y}(t)$	Time history of loads
$\mathbf{y}^\dagger$	Length M vector of maximum absolute loads their across time

	history
$\mathbf{Y}_k$	$N \times M$ array containing N length M vectors of loads corresponding to N design vectors
$\mathbf{Y}_k$	$N \times M$ array containing N length M vectors of loads for load case indexed k, corresponding to N design vectors
$\mathbf{y}_k$	Vector of loads for load case indexed k
$\mathbf{Z}$	$N \times D$ array of N design vectors in D dimensions
$\mathbf{z}$	Vector describing a parametrised aircraft design
$\mathcal{D}$	Data/dataset; tuple of observed points, $\mathbf{X}$, and corresponding responses, $\mathbf{y}$
$\mathcal{X}_c$	Set of critical load cases
$\text{diag}(A)$	Leading diagonal of A
$\text{tr}(A)$	Trace of A
ν	Matérn kernel smoothness parameter
Ω	Inverted covariance matrix
$\overline{\mathbf{X}}$	Array of M sampled argmaxes
$\overline{\mathbf{x}}$	Sampled argmax
$\overline{\mathbf{y}}$	Vector of enveloping loads; the maximum absolute loads for each IQ
$\Phi(\cdot)$	Random feature approximation
$\phi(\cdot)$	Scaling function
$\phi^{-1}(\cdot)$	Rescaling function
ρ	Vector of anisotropic kernel lengthscales
ρ_i	Anisotropic kernel lengthscale in dimension i
τ	Anisotropic scaled euclidean distance
θ	Vector of hyperparameters
$\triangleq$	Definition; is defined as
ε	Standard deviation of Gaussian noise
ϑ	Vector of marginal distribution parameters
$\widehat{R}$	Scalar potential scale reduction factor
$A \diamond B$	Kronecker product between A and B

$A \otimes B$	Tensor product between A and B
$a(\cdot)$	Acquisition function
$a^\intercal$	Transpose of a
$A_{/i}$	Matrix or set A with the i^{th} entry omitted
$c(\cdot, \cdot)$	Correlation kernel (covariance kernel with unit marginal standard deviation)
D	Integer number of index dimensions
$f(\cdot)$	External force (loads) calculation process
$g(\cdot)$	Internal force (stress) calculation process
$h(\cdot)$	Maximum absolute loads across time history approximation process
i	Subscript to denote position
K	Integer number of discrete load cases
$k(\cdot, \cdot)$	Covariance kernel
K_{AB}	Array obtained by evaluating covariance kernel between arrays A and B
l	Isotropic kernel lengthscale
$l(\cdot)$	Simplified loads process
$l_{\mathbf{z}}(\cdot)$	Loads process given fixed design vector
$l_k(\cdot)$	Approximate loads process for a specific load case indexed k
M	Integer number of IQs under consideration (loads), sampled maxima (critical case identification), or sampled design vectors (stochastic loads)
$m(\cdot)$	Mean function
m_A	Vector obtained by evaluating mean function over array A
N	Integer number of observations
R	Integer number of compression dimensions
r	Scalar isotropic scaled euclidean distance
S	Integer number of samples
T	Simulation budget / maximum number of iterations
t	Scalar time (loads) or integer iteration number (optimisation)

u	Scalar tolerance parameter
w	Weights
Y	Random vector, uncertain loads
y	Scalar response (abstract) or a single interesting quantity (loads)
Z	Random vector, uncertain design vector

Sets:

$\mathbb{R}$	Real numbers
$\mathbb{R}^D$	D-dimensional vector of real numbers
$\mathcal{X}$	Space of all load cases under consideration
$\mathcal{Z}$	Space of all (derivative) designs under consideration
Θ	Space of all permissible hyperparameters

Probability Distributions:

$\mathcal{GP}(m, k)$	Gaussian process with mean function m and covariance kernel k
InvGamma(a, b)	Inverse Gamma distribution with parameters a and b
MVN(μ, Σ)	Multivariate Normal distribution with mean vector $\mathbf{m}$ and covariance matrix Σ
Normal(σ, σ)	Normal distribution with mean μ and standard deviation σ
Uniform(a, b)	Uniform distribution between a and b

Probability and Statistics:

$\hat{a}$	Approximation/surrogate for a
$\mathbb{E}$	Expectation
$\mathbb{V}$	Variance
$\mathcal{L}(\cdot)$	Likelihood
Md$(\cdot)$	Sample median
μ	Population mean
Π	Arbitrary cumulative density function
π	Arbitrary probability density function
Σ	Covariance matrix
σ	Population standard deviation

$a\tilde{b}$	a is distributed according to b
a^*	Prediction of a
$p(\cdot)$	Probability

Chapter 1

Introduction: Data, Surrogate Models, and Uncertainty

1.1 On Data

Modern engineering design is experiencing a rapid shift towards digitisation. The complete lifecycle of product development – whether this be aircraft, turbomachinery, automobiles or household appliances – is increasingly simulated, with complex simulations of real world physics used to assess, predict and/or monitor various performance characteristics under a wide range of operating conditions. This digital transformation has occurred to such an extent that some researchers have adopted the term *Industry 4.0* [1].

Means of acquiring performance information for each and every subsystem of the product has spurred substantial interest in utilising it. The opportunity to analyse the interaction between subsystems previously analysed in isolation has not been overlooked, and industry and academic interest in multidisciplinary design has surged in recent years, encouraged by (relatively) recent capability to acquire and process larger and larger amounts of data via increasingly realistic and affordable simulation.

Simultaneously, advances in the fields of computer science and computational statistics have provided scientists and engineers with increasingly sophisticated tools to process and analyse larger quantities of increasingly complicated types of data. In particular, the potential to analyse complex systems of many interacting variables at the lowest level – made possible by the capability to both acquire and load this data into memory (crucially, on relatively inexpensive hardware) – has attracted considerable interest from the engineering design community due to potential applications in multidisciplinary design.

There are, however, obstacles towards adoption of many Machine-Learning (ML) derived methods seamlessly into the industrial engineering design process: born primarily of increasingly affordable means of generating and storing information, many modern ML algorithms function most effectively on either large volumes of data (high dimensional regression and/or classification, deep learning) or in situations permitting relatively inexpensive data acquisition (many-objective heuristic opti-

misation in general). In engineering design applications, and certainly within the context of high-value manufacturing – which in particular requires high precision, detailed calculations of complex physics – such acquisition rates and/or volumes of data are unprecedented, even after accounting for algorithmic improvements and advances in computational power. A significant barrier towards effective, industrialised Multidisciplinary Design Optimisation (MDO) – which by its nature requires cross-disciplinary correlations between design variables to be digested and understood – is consequently the provision of data at sufficient speeds and/or quantities to make this possible.

1.2 On Surrogate Models

The current generation of physics models are often referred to as "expensive" due to the relative amount of resources required to query them. This cost can be expressed in any number of ways, usually in terms of clock-time, but is in general "fixed" by that particular model: for example, while provision of faster clock time is often possible through purchase of more powerful hardware or via cloud based services, the cost of the model then manifests as a monetary expense associated with the rent and/or upkeep of these resources instead.

A physics based simulation – herein a *simulator* – can be considered more generally as a means of collecting information, analogous to performing a physical experiment. One can consider that testing a component in a wind-tunnel and carrying out a Computational Fluid Dynamics (CFD) simulation of the same component might seek to answer similar or identical questions about that component's performance characteristics. Indeed, the introduction of accurate simulation to engineering design provided means of analysing a much larger number of design configurations, thus permitting more rigorous exploration of a given design space by reducing the cost associated with data acquisition relative to physical testing.

Current barriers towards multidisciplinary optimisation can be tackled in an analogous fashion. Viewed as a "black-box" (that is, a function which we cannot "look

inside"), all simulators act as functions, transforming some abstract "input" to a set of quantities of interest (QOIs) or "outputs". Assuming this input can be parametrised, this input-output relationship can be expressed in a neat, mathematical expression:

$$\mathbf{y} = f(\mathbf{x}) \tag{1.1}$$

The apparent simplicity of equation (1.1) is due to the abstraction of the context surrounding the simulator, f; its inputs, $\mathbf{x}$; and its outputs, $\mathbf{y}$. Of importance is that nothing is necessarily absent from this expression, provided all we care about is the value of $\mathbf{y}$ given a specific input $\mathbf{x}$. It is this fact that provides a solution to the data-dilemma facing current generation simulators: by suitable parametrisation of the model inputs, one can consider learning the input-output relationship described by (1.1), hence approximating it with a "purely mathematical" alternative:

$$\mathbf{y} = f(\mathbf{x}) \approx \hat{\mathbf{y}} = \hat{f}(\mathbf{x}) \tag{1.2}$$

$\hat{f}$ is usually referred to as a meta-model or *surrogate*. $\hat{f}$ is assumed to have no *explicit* knowledge of any physics, in that it does not solve or consider systems of physics based equations: it simply seeks to represent the relationship between $\mathbf{x}$ and $\mathbf{y}$. In this sense, a surrogate model is simply a regression of $\mathbf{y}$ onto $\mathbf{x}$, and is "purely mathematical" in the sense that it is agnostic to both what goes on inside the original black-box, f, and even what the original inputs and outputs actually represent. Absence of such considerations make a surrogate significantly cheaper to query than a simulator, as no effort needs to be spent ensuring large systems of state equations or the like are satisfied.

Effort is required to construct a surrogate model in the first place, given the relationship between $\mathbf{x}$ and $\mathbf{y}$ must be "learned" from data. This requires samples from the simulator, available either via some database of existing results, or – perhaps more frequently – via running the simulator at a selection of chosen sample points

by performing a Design Of Experiments (DOE). Given that this involves multiple queries to a potentially expensive computer code, this represents the bulk of the cost associated with the use of a surrogate model.

Given this description, surrogate modelling may seem like a canonical *supervised learning* problem – and in many respects it is. However, given that we require a surrogate model precisely because the target function is expensive, this problem is often "data sparse". The outputs of surrogate models are also frequently fed into other "downstream" processes, meaning it is critically important to understand the performance of the model, and in particular the effect of any differences between the surrogate and the target function on other components in the analysis process.

1.3 On Predictive Uncertainty

The task of identifying the relationship between inputs, $\mathbf{x}$, and outputs, $\mathbf{y}$, using a finite number of samples, is one of inference, and it is extremely rare that this functional relationship can be identified exactly outside of trivial examples.

This has two important consequences. The first, and most obvious, is that equivalence between f and $\hat{f}$ can only be obtained in artificially contrived examples. Practically, this implies that we should not expect $f(\mathbf{x}) = \hat{f}(\mathbf{x})$ for all possible $\mathbf{x}$, unless f is a very simple function (in which case we do not require a surrogate!). The second, and perhaps more subtle point, is that it may not be possible to identify a unique model for f from a finite number of samples. Both of these points contribute to uncertainty associated with the use of the surrogate in place of f, i.e., there will be an unknown error associated with the use of a surrogate model. This is typically referred to as *predictive uncertainty*, so called because predictions made by the model ought not to be considered deterministic, but rather distributions of possible outcomes.

To expand on this further, consider a surrogate model, $\hat{f}$, for a simulator, f. The

relationship between the two can be expressed as follows:

$$f(\mathbf{x}) = \hat{f}(\mathbf{x}) + \varepsilon_{\mathbf{x}} \tag{1.3}$$

where $\varepsilon_{\mathbf{x}}$ is the error (or residual) between the surrogate and the simulator at $\mathbf{x}$. For a set of N points $\{\mathbf{x}_1 \dots \mathbf{x}_N\}$, it follows that we have a corresponding set of N residuals. The distribution of residuals is a summary of errors on a particular dataset. However, what we are interested in is the performance of the surrogate at points at which $f(\mathbf{x})$ is *not* available, and thus exhibit an error that is presently unknown. The best way we can accomplish this is by reasoning with the possible distribution of error, i.e., to assign some measure of confidence to all values of error we view as plausible. There are a multitude of frameworks for accomplishing this, the most widely applied and understood being the theory of probability. A brief introduction to probabilistic modelling is provided in appendix A.

1.4 On Aircraft Flight Loads

Detailed structural optimisation (or sizing) is amongst the most important stages of industrial aircraft design, and can contribute significantly to design performance *and* program time. This is a challenging and time consuming task: Iorga et al. [2] put the number of design variables at well over 2,000 and the number of individual structural constraints at over 100,000 for just the design of a wing-box. Accurate prediction of the loads, which are responsible for determining the stresses against which the structure is sized, is consequently of paramount importance.

Precise calculation of such loads is a computationally intensive procedure, and may require several different analysis procedures depending on the type of loading [2]. The design process has historically been progressed from early stages by using fast, coarse approximations, with full, detailed loads calculations reserved for resolving the finer details of the design and for certification. However, relatively recent interest in making robust design decisions earlier in the design process – motivated by the potential financial consequences of prematurely committing to certain configuration

choices – has encouraged use of certification standard loads simulation early on in the design cycle. While theoretically providing a reduction in uncertainty due to providing considerably more accurate predictions of the loads, ability to query the loads process only a relatively limited number of times due to computational expenses leads to a set of different uncertainty quantification related challenges.

1.5 Aims and Objectives

The formal objectives of this work are stated as follows:

1. A demonstration of the use of surrogate models for providing rapid approximations of the aircraft loads analysis process, with examples of industrially relevant use-cases utilising such models.

2. A critical assessment of the uncertainties present at the loads-structural coupling interface, investigating the consequences of the use of an expensive, detailed model early in the design process at stages where the structure does not have detailed definition.

3. Development of models capable of representing the uncertainties described by item 2.

4. A robust, industrialisable method for constructing models of uncertainty, to facilitate the above.

1.6 Outline of This Thesis

This work adopts a probabilistic view of surrogate modelling and by doing so, facilitates an intuitive understanding of many types of uncertainties implicit with the process.

Through Gaussian Process regression, introduced in chapter 2, we frame surrogate modelling as a task of inference over types of functions, and in chapter 3, discuss different strategies for performing such inferences, along with their implied assumptions.

Chapter 4 discusses loads-structural coupling, an industrial multidisciplinary modelling problem applicable to aircraft design. A case study is presented which makes use of considerations developed during chapters chapters 2 and 3, and also serves as an introduction the applied problem of modelling aircraft loads. Chapter 4 concludes with an analysis of uncertainty present in the procedure.

Motivated by the problems described at the end of chapter 4, chapter 5 discusses development of an *industrialisable* method for constructing robust models of uncertainty (or *emulators*) applicable to general, data sparse surrogate modelling problems, using considerations outlined in chapters 2 and 3. The performance of such models is evaluated on a variety of test functions in chapter 6.

In chapters 7 and 8 we revisit the aircraft loads process and address a major source of uncertainty: the selection of the so called "critical" aircraft load cases. Using the methods for constructing emulators developed in chapter 5 and validated in chapter 6, we first approach two slightly simplified versions of the problem in chapter 7, before attempting to tackle the full problem in chapter 8.

1.6.1 Contributions to Knowledge

This work contributes several developments, both in terms of methods and application of theory.

The surrogate modelling techniques documented in the case study introduced in chapter 4 have been applied to both preliminary sizing of an aircraft structure [2] and for propagating correlated structural design uncertainties [3].

Application of the theory for constructing robust emulators introduced in chapter 5 for the purposes of surrogate modelling is novel, as are the comparisons between maximum likelihood estimation and fully probabilistic treatments of the models documented in chapter 6.

Both facets of the critical load case selection as presented in chapter (8) constitute advancements in capability: the use of Bayesian Optimisation for critical case identification is novel; and the probabilistic load case downselection strategy provides

previously unrealisable industrial capability.

Finally, the mathematical contextualisation of the loads process featuring design variability as a stochastic process in chapter (8) is original, as is the attempt to model it as such.

1.6.2 Appendices

Several appendices are provided to supplement the main text, and will be referenced where necessary if a terse explanation of a method or concept is required in order not to detract from a more salient point.

Optional preliminaries required to appreciate probabilistic modelling in an intuitive manner are introduced in appendix A, and an introduction to the basic intuitions behind GP regression can be found in appendix B, along with a mathematical descriptions of the basic algebraic manipulations required for practical use. Several useful extensions to the standard GP regression model are documented in appendix C.

Supplementary material for the case studies in chapters 4 and 6 is contained in appendices D and F, respectively, and Appendix G contains a more detailed overview of Bayesian Optimisation strategies to supplement chapter 7.

Code

A fully featured python package implementing all of the methods discussed in this document can be found online. Please contact the author for access, as the repository is private.

Chapter 2

Practical Gaussian Process Regression

We discuss Gaussian Process regression in the context of applied problems, reviewing the strengths of the method, the process of selecting a covariance kernel and mean function, and the types assumptions such choices contribute to the model.

2.1 An Introduction to GP Regression

Gaussian Processes (GPs) are a popular choice for performing non-parametric regression as they invoke a mathematically convenient probability distribution over functions. This chapter adopts a Bayesian view of this procedure, in which a GP defines a prior distribution over different functions considered candidate explanations for observed data. Readers unfamiliar with Bayesian statistics are advised to read the brief introduction contained in appendix A.

By conditioning this distribution over functions on a particular set of data, the resulting conditional distribution is said to describe a subset of "random functions" which are capable of "explaining" (or are *compatible with*) these observations. GP regression can thus be interpreted as a technique for attempting to infer the mechanism by which some "inputs", $\mathbf{x}_1, \mathbf{x}_2 \ldots$, are mapped to corresponding "outputs", $y_1, y_2 \ldots$, given some assumptions. We may think of these different possible functions as "(data) generating mechanisms", "candidate explanations" or "models for the data".

GPs are probabilistic models of models, in the sense that one obtains a prediction in the form of a probability distribution over different ways the observed data could have been generated. Ignoring for the time being how this distribution comes about, from a surrogate modelling perspective this is convenient as a GP can be said to quantify its own predictive uncertainty: *all* the models that are compatible with the data and the assumptions of the GP are used to formulate a prediction. GPs used as predictive models are occasionally referred to as *emulators* for this reason [4], as they are models of both what is *and is not* known about the function they are intended to provide a representation of.

How good of a representation this is primarily depends on the choice of the GP's parameters, and how the assumptions implied by these parameters align with those of the simulator the GP should provide a model for. This chapter introduces standard considerations applicable to most GP regression problems.

For an accessible introduction to the intuition behind GP regression, along with a

more detailed description of the basic mathematical manipulations required to use the method in practice, readers are encouraged to read appendix B. Rasmussen and Williams [5] provide a detailed, technical reference. In engineering literature, GP regression is often referred to as *Kriging*.

2.1.1 Basics

A GP is a probability distribution defined over a D-dimensional space of input variables referred to as the index. The index corresponds to the space of input variables for the function one wishes to model as a GP. A single point in the index is a D-dimensional vector, in this document denoted by $\mathbf{x}$, unless specified otherwise.

All GPs have two parameters which influence their behaviour, both of which are functions of the index: the mean function, $m(\cdot)$; and covariance kernel, $k(\cdot, \cdot)$.

Let $\mathbf{y}$ be a length N vector of responses from a function we wish to model, assumed to have been observed at a set of N sample points which are arrange in an $N \times D$ array, $\mathbf{X}$. To write $\mathbf{y}$ as being distributed according to a GP, one writes:

$$p(\mathbf{y}|\mathbf{X}) \sim \mathcal{GP}(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X})) \tag{2.1}$$

where the somewhat abusive (yet fully conventional) notation $m(\mathbf{X})$ implies evaluating $m(\cdot)$ for each of the N points in $\mathbf{X}$ to produces a length N vector, $\mathbf{m}$; and (again, somewhat abusively) $k(\mathbf{X}, \mathbf{X})$ implies evaluating $k(\cdot, \cdot)$ pairwise between the same N points to produce an $N \times N$ covariance matrix, $\mathbf{K}$.

Equation (2.1) refers to an unconditional GP, as it simply expresses the probability of observing the responses, $\mathbf{y}$, for a given $\mathbf{X}$, assuming the chosen mean function and covariance kernel.

Predictions are made by conditioning a GP distribution on data, represented by a tuple, $\mathcal{D} \triangleq \{\mathbf{X}, \mathbf{y}\}$, which contains the sample points and responses. Intuitively, this restricts the distribution of the GP to functions which are compatible with the observations in $\mathcal{D}$.

The predictive distribution at a point of interest, $\mathbf{x}^*$, conditional on $\mathcal{D}$, is denoted:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) \sim \mathcal{GP}(m^*(\mathbf{x}^*|\mathcal{D}), k^*(\mathbf{x}^*|\mathcal{D})) \tag{2.2}$$

where y^* represents the (unknown) response at $\mathbf{x}$; and $m^*(\mathbf{x}^*|\mathcal{D})$ and $k^*(\mathbf{x}^*|\mathcal{D})$ represent computation of the conditional mean, $\mathbf{m}^*$, and covariance matrix, $\mathbf{K}^*$, respectively, where:

$$\begin{aligned} \mathbf{m}^* &= m(\mathbf{x}^*) + k(\mathbf{x}^*, \mathbf{X}) \cdot [\mathbf{K}]^{-1} \cdot (\mathbf{y} - \mathbf{m}) \\ \mathbf{K}^* &= k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{x}^*, \mathbf{X}) \cdot [\mathbf{K}]^{-1} \cdot k(\mathbf{X}, \mathbf{x}^*) \end{aligned} \tag{2.3}$$

For a more thorough explanation please see appendix B.

2.1.2 Assumptions

GP regression, in it's standard implementation, assumes that the distribution of all "possible explanations" for a dataset compatible with the GP's parameters is jointly Gaussian. In simple terms, this is equivalent to asserting that the data is a draw from something which behaves like a multivariate Gaussian distribution.

Adopting a Bayesian view, this need not be interpreted as the true data generating mechanism (whatever it may be) *definitely* following a multivariate Gaussian distribution: this assumption is simply a modelling decision used for the algebraic convenience of invoking a prior probability distribution over functions that can be conditioned on data. However, the strength to which this assumption applies can have consequences on how the resulting model may be interpreted. The consequences of how strongly this assumption holds in terms of predictive performance are investigated by Scheuerer et al. [6], and described in [5, ch.2 sec.4]. In general, the predictive mean of a GP is the best unbiased predictor in the mean-squared sense if the loss function associated with an incorrect prediction is symmetrical, that is; if the penalty associated with making a prediction that is either too large or too small

by the same amount is equal.

We note critically that theoretical optimality is not required for the model to be useful. Transformation of a GP to obtain a stochastic process with non-gaussian marginals has merit in select applications and is discussed briefly in appendix C.4.

Further assumptions about the properties of the data generating mechanism are encoded by the GPs parameters, and are the subject of the remainder of this chapter.

2.2 Mean Function

Choice of mean function (sometimes termed the "trend") is less critical of a modelling assumption relative to the kernel, but still confers several important assumptions.

Use of a zero mean function implies all of the variation observed in the data is modelled by the GP. This is beneficial as it ensures all assumptions about the data are encoded by the covariance kernel, and is canonical in the machine-learning literature discussing GP regression [5, ch. 2] [7, ch.15].

Use of a parametrised mean function (i.e. some function of the index) implies a regression of the inputs using the chosen mean function, with the GP used as a model for the residuals. This can provide better behaviour concerning long-range predictions but can be prone to introducing over-parametrisation for simple problems if the choice of mean function is arbitrary [8].

In general, if the data exhibits a known mean, this is useful information to include, though makes reasoning with the desired properties of the covariance kernel a more nuanced and problem specific task: any overlap in the modelling properties of the kernel and the mean potentially leads to over-parametrisation.

2.3 Covariance Kernel

The choice of covariance kernel encapsulates the most important modelling assumptions made when performing GP regression. The choice of kernel determines the subset of functions over which the GP invokes a probability distribution (see appendix B), and consequently the properties of the resulting regression model.

The ability to choose this subset of functions via the kernel is a powerful modelling tool. Given that GP regression is a task of probabilistic inference (specifically, we are seeking to infer candidate "explanations" for observed data), ability to restrict candidate functions to those with properties the data is known or expected to exhibit can significantly improve the quality of the model, both in terms of predictive accuracy and reliable quantification of predictive uncertainty. This is especially important for sparse data and/or uncertainty quantification focused applications in which the data alone may not reveal much (or anything) about the properties of the data generating mechanism, and one might wish to impose structure on the space of possible models that the data is incapable of revealing by itself. An important example of such structure which we will explore shortly in section 2.3.6 is smoothness. This is the principle advantage of GP regression relative to other forms of non-linear regression such as neural-networks, which typically require significantly larger quantities of data to function reliably.

It is worth noting that while the properties that the data generating mechanism does exhibit may often be unknown or even unknowable, knowledge of properties that it does not exhibit can often be deduced from specifics of the problem: this is equivalently useful information.

2.3.1 Hyperparameters

Most covariance kernels, in addition to accepting a pair of points from the index, contain extra parameters which influence the behaviour of the kernel. In the context of GP regression, in which the covariance kernel is itself a parameter, these are referred to as *hyperparameters* (literally, "parameters of parameters").

We assume the hyperparameters of an arbitrary kernel, k, are summarised by a vector, θ, which is assumed to contain all relevant hyperparameters for that particular kernel. A specific kernel "given" a vector hyperparameters is denoted $k(\mathbf{x}, \mathbf{x}'|\theta)$, in order to emphasise the dependency of k on θ.

Identifying the appropriate hyperparameters for a particular kernel and dataset is one of the central tasks when performing GP regression, and is usually what is implied when one discusses "fitting" a GP to a dataset. Selection of an appropriate *type* of covariance kernel, however, is an essential first step, and can even determine the most suitable manner in which the hyperparameters should be identified.

2.3.2 Marginal Standard Deviation

Almost all kernels are scaled by an amplitude (sometimes "bandwidth") hyperparameter, α, which controls the (unconditional) marginal standard deviation of the GP, roughly equating to the magnitude of the oscillations in the functions the GP encodes a distribution over.

For example, if $c(\mathbf{x}, \mathbf{x}')$ is a function describing correlations (that is, *normalised* covariances) between points, then one can obtain a covariance matrix by scaling by α^2:

$$k(\mathbf{x}, \mathbf{x}|\theta) = \alpha^2 c(\mathbf{x}, \mathbf{x}') \tag{2.4}$$

where θ is a vector containing the amplitude hyperparameter, α, along with any other parameters used by $c(\mathbf{x}, \mathbf{x})$, i.e., $\theta \triangleq [\alpha \dots]$. A visual demonstration of the effect of α is shown for a 1D function in figure (2.1).

The marginal standard deviation hyperparameter is important as it influences the sensible ranges of the other kernel hyperparameters, particularly the kernel length-scale(s), which will be introduced shortly.

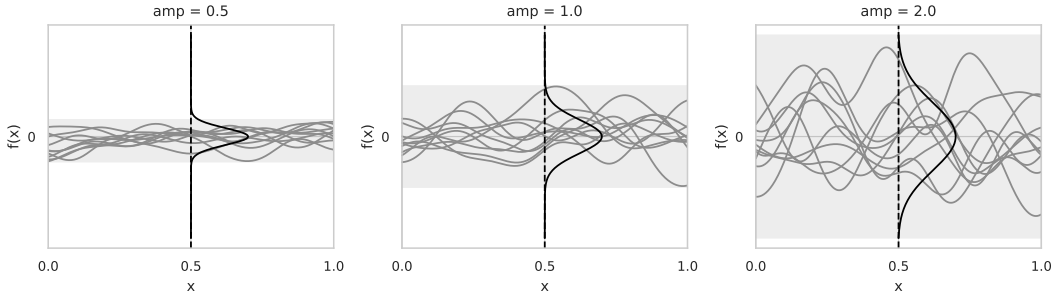

Figure 2.1: Visualisation of changing the marginal standard deviation ("amplitude") hyperparameter. The shaded areas correspond to three standard deviations. Unscaled marginal distributions are sketched at $x = 0.5$.

2.3.3 Stationary Kernels

The canonical class of kernels in the GP literature are those referred to as stationary. A simple description of a stationary kernel is that the covariance designated by such kernels is translation invariant over the index on which they are defined, such that:

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} + \Delta, \mathbf{x}' + \Delta) \tag{2.5}$$

where Δ is some perturbation.

This implies that the covariance between any two points in the index as determined by a stationary kernel is a function of only the distance between those two points, and that the abstract "properties" of the sample functions encoded by such a kernel are constant over the index. An example of such a property might be the frequency of a periodic function. In contrast, if the properties of these functions were to be different depending on the absolute position in the index at which they are observed, a kernel encoding such properties would *not* be stationary. A visual example is shown in figure (2.2): the sample paths shown on the non-stationary plot can be observed to become more oscillatory as $x \to 1$, while those on the stationary plot oscillate at a fixed frequency regardless of where in the index one looks. A formal mathematical description of the stationary property can be found in [5, ch. 4].

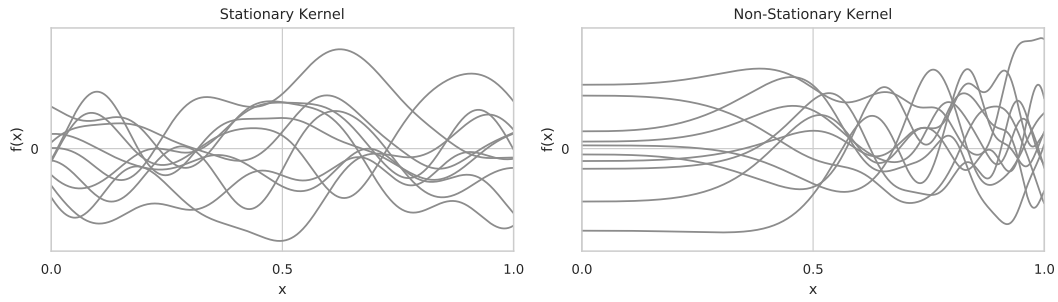

Figure 2.2: Sample paths drawn from stationary (left) and non-stationary (right) kernels.

2.3.4 Lengthscales

Many popular stationary kernels are functions of a scaled euclidean distance:

$$k(\mathbf{x}, \mathbf{x}') = \alpha^2 c(r) \tag{2.6}$$

where $c(r)$ determines the correlation between two points as some function of the scaled distance between them, given by:

$$r = \frac{(\mathbf{x} - \mathbf{x}')^2}{l^2} \tag{2.7}$$

The (strictly positive) parameter, l, is a hyperparameter known as a lengthscale. A smaller lengthscale causes r to increase more rapidly with distance, which translates to more quickly decaying covariance as a function of the absolute distance between points. It is worth observing that l is a constant, and hence implies the assumption that functions encoded by a particular stationary kernel have the same lengthscale of l everywhere in the index.

In terms of the properties of functions encoded by a stationary kernel, the lengthscale can be interpreted to control the degree of nonlinearity over which the kernel invokes a distribution [9]. Short lengthscale kernels are said to be more nonlinear, as quickly decaying covariance between points implies more "local" variation (with identical amounts of local variation assumed to be present elsewhere, as implied by the stationary property). Functions drawn from long lengthscale kernels, on the other hand, become asymptotically linear as $l \to \infty$, as points far from one another

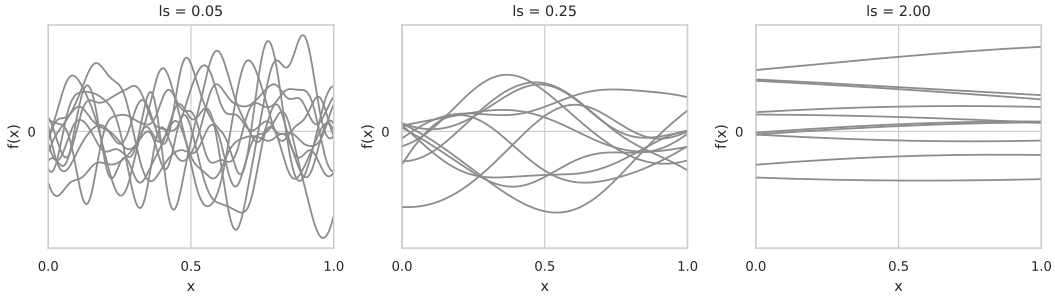

Figure 2.3: Sample paths drawn from stationary GPs with different lenghtscale kernels.

can retain high covariance. A visual example is provided by figure (2.3).

2.3.5 Isotropy and Automatic Relevance Determination

The scaled distance described by equation (2.7) is referred to as *isotropic*, since a single lengthscale is shared for each dimension of the index. Intuitively, if $k(r)$ is a stationary, isotropic kernel defined over a D-dimensional index, then the subset of functions over which k invokes a distribution will exhibit identical nonlinearity in all D input dimensions. Since this is an unrealistic assumption for many practical problems (particularly in engineering, where model inputs tend to represent quantities which may have different units and/or scales) it is more common to use an *anisotropic* scaled distance, τ:

$$\tau = \sum_{i=1}^{d} \frac{(x_i - x_i')^2}{\rho_i^2} \tag{2.8}$$

where x_i represents the i^{th} element of either $\mathbf{x}$ or $\mathbf{x}'$. Anisotropic kernels are are parametrised by a vector of D dimensionwise lengthscales, $\rho \triangleq [\rho_1 \dots \rho_D]$, controlling the relative nonlinearity in each of the input dimensions independently. Kernels which are functions of τ are occasionally referred to as implementing Automatic Relevant Determination (ARD) [see 10], as some researchers interpret the magnitude of ρ_i for a given dimension to correspond inversely to that dimension's perceived "importance".

Finally, it deserves mentioning that there is some flexibility in terms of how one

Name	*Notes*
Exponentiated Quadratic	Infinitely smooth
Matérn (full)	Smoothness controlled by ν
Matérn ($\nu = 5/2$)	Twice differentiable functions
Matérn ($\nu = 3/2$)	Once differentiable functions

Table 2.1: Common stationary kernels and their corresponding smoothness properties.

parametrises the scaled distance described by equation (2.8), with some authors preferring to use squared and/or inverted versions of the lengthscales, ρ.

Anisotropic kernels are canonical for almost all engineering surrogate modelling applications using GP regression.

2.3.6 Encoding Smoothness

In many cases, we wish to model functions which are smooth. A somewhat informal description of this property is that we expect $f(\mathbf{x})$ to be close to $f(\mathbf{x} + \delta)$, where δ is a small perturbation: i.e., we do not anticipate that moving a short distance anywhere in the index will produce a large change in the value of the function we are interested in modelling. A function which is not smooth, for comparison, might contain one or more discontinuities, such that the previous does not hold everywhere in the index. A rigorous mathematical treatment of sample function smoothness can be found in [11].

For applied problems, the "degree" of smoothness encoded by a given kernel is often expressed in terms of the number of times a sample path from such a kernel can be differentiated with respect to the index. This is convenient for practitioners, as it is often a reasonably straightforward property to reason with: when building surrogate models of computer physics codes, for instance, the properties of the physical quantities one wishes to create a model in terms of may bound the number of times one might expect the output to be differentiable with respect to the index.

Table 2.1 contains the smoothness assumptions for several popular stationary covariance kernels. The Exponentiated Quadratic (EQ; occasionally "squared exponential" or "gaussian") kernel is ubiquitous in the scientific literature, despite encoding

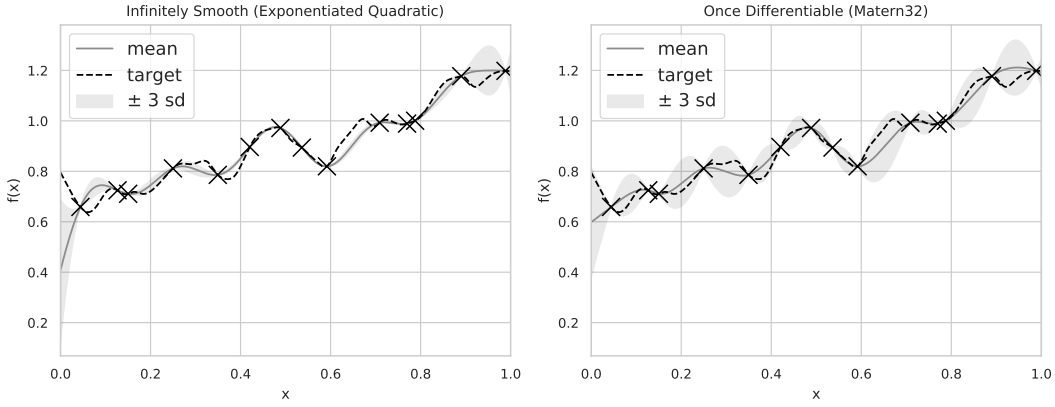

Figure 2.4: Posterior distributions for an infinitely smooth (left) and once differentiable (right) kernel used to model a once differentiable 1D function (dashed line).

overly aggressive smoothness assumptions for many problems. This is due to its (relative) simplicity, and the fact that it is universal (see [12]), which in practical terms implies the kernel can model any arbitrary continuous function given enough data. This makes the EQ kernel a reasonable first choice for many modelling problems, though the caveat of requiring a sufficient density of observed points to produce a quality model of any continuous function should be emphasised.

Failure to sufficiently match the smoothness properties of the target function can have important modelling consequences. Figure (2.4) provides a simple 1D example on a synthetic function, demonstrating harmful overconfidence of the EQ model on a target function which is only once differentiable (only the first derivative is finite). In several regions of the index the infinitely smooth kernel exhibits pathological overconfidence, manifesting as certain values of true function being designated improbable by the model. With reference to the previous paragraph, it is worth noting that the EQ model is capable of approximating this synthetic function, but not without a larger volume of data.

2.4 Limitations

The most significant disadvantage of GP regression relative to other surrogate modelling techniques is the implied computational cost. Fitting a GP to data (to be discussed in the next chapter) requires repeated inversion of the GP covariance ma-

trix, which is a has a computational complexity of $\mathcal{O}(N^3)$. The scalability of GP regression in its standard implementation is consequently poor, and it is hard to recommend using a GP for over approximately a few hundred samples, depending on hardware. However, due to the convenient theoretical advantages of GP regression, recent attempts have been made to adapt it to large sample sizes. Wilson et al. [13] reviewed some of these methods.

A further, under-recognised limitation with GP regression is scaling with respect to the number of input dimensions. A GP covariance matrix can be considered to be a function of the distance between points, and this distance increases exponentially (all else remaining equal) with each additional dimension. High numbers of input dimensions consequently imply larger distances between points, to the extent that new points are "further away" from existing observations. While it depends on the quantity of available data, this somewhat limits the effectiveness of GP regression in its default implementation through interaction with its first and primary limitation: beyond more than *approximately* 15 input dimensions, a large volume of data is required to ensure sufficient coverage of the input space. Without addressing sample size scalability, such sparse coverage will result in predictions that are almost always distant from observed data (in the context of the kernel), thus exhibiting large amounts of predictive uncertainty and tending towards the unconditional mean of the model.

Finally, the effectiveness of GP regression for a given problem depends strongly on how the assumptions encoded by the kernel align with those of the real data generating mechanism.

2.5 Summary

We briefly introduce GP regression, a technique for performing inference over functions. By conditioning this distribution on observations from some data generating mechanism, one obtains a distribution over only those functions compatible with the assumptions encoded by the GP and the supplied observations. This condi-

tional distribution can be used to provide statistical predictions of the value of the hypothesised data generating mechanism at new points.

The textbook of Rasmussen and Williams [5] is a valuable resource for any practitioner using GP regression. Duvenaud [14, ch.2] presents a detailed exposition on encoding structure with different kernels in the context of predictive modelling, and Gelman et al. [15, ch.21] discusses GP regression in a probabilistic modelling context.

Surrogate models such as GP regression are a mainstay of the modern engineering design process, and are put to use for an increasingly diverse array of tasks. Viana et al. [16] provide a comprehensive review of surrogate modelling since its inception. Forrester and Keane [17] reviewed applications in the context of optimisation, and Iooss and Lemaître [18] reviewed sensitivity analysis.

Chapter 3

Hyperparameter Inference for Gaussian Processes

A key contribution to the uncertainty associated with the use of a surrogate model is inference of the parameters which fit it the model to a particular set of data. In this chapter, we introduce some different methods of fitting such models to data, discuss their assumptions, and provide a basic theoretical justification for the *robust* models to be introduced later.

3.1 Fitting a GP to Data

Assuming that a set of simulator outputs can be modelled by a Gaussian Process is equivalent to assuming that the vector of observations, $\mathbf{y}$, is a draw from a specific "data generating mechanism" which can be modelled by a GP. Identification of data generating mechanisms compatible with a given dataset and GP prior is what is intuitively meant by "fitting" a GP. Given a choice of mean function and kernel, as discussed in the previous chapter, this requires inference of the model hyperparameters.

The hyperparameters for a given GP are seldom known in advance, and must be inferred, given data. This implies, given a finite number of samples, that the hyperparameters are a random variable, described by some conditional probability distribution describing the compatibility of the hyperparameters with the data and any prior assumptions. An expression for this distribution can be obtained via *Bayes' rule* (see appendix A.3.1).

Let the hyperparameters of the GP be represented by θ, and the data by $\mathcal{D}$. We assume additional information about θ (which we will elaborate on in section 3.3.2) is encoded by a prior distribution, π_θ. The posterior distribution of θ is then described by:

$$p(\theta|\mathcal{D}, \pi_\theta) = \frac{p(\mathcal{D}|\theta)p(\theta|\pi_\theta)}{p(\mathcal{D})} \tag{3.1}$$

Computing $p(\theta|\mathcal{D}, \pi_\theta)$ via equation (3.1) analytically is not usually tractable, so it must be approximated, and the chosen method of approximation contributes different assumptions to the resulting model. This, in turn, influences the subset of functions over which the GP invokes a distribution as a result. How one performs hyperparameter inference should thus be considered carefully as it can significantly influence the quality of the resulting model.

In this chapter, we will explicitly formulate hyperparameter identification as a probabilistic model before introducing two classes of methods that can be used to approx-

imate the posterior: point predictions, which summarise the posterior by a single point; and approximations to the entire distribution, which take the entire shape of the distribution into account.

3.2 Probabilistic Model

We wish to identify the posterior distribution of the hyperparameters, θ; given data $\mathcal{D} \triangleq \{\mathbf{X}, \mathbf{y}\}$; and a prior distribution, π_θ, assuming that $\mathbf{y}$ is modelled as a draw from a GP with kernel k and mean function m.

The likelihood of a specific vector of hyperparameters given the data is equivalent to the probability of the data given those specific hyperparameters multiplied by the probability of those hyperparameters as determined by their prior. Let $p(\mathbf{y}|\mathbf{X}, \theta)$ be the probability of the data conditional on the hyperparameters and $p(\theta|\pi_\theta)$ be the probability of the hyperparameters. The probability of the data given those hyperparameters is:

$$p(\mathbf{y}|\mathbf{X}, \theta, \pi_\theta) = p(\mathbf{y}|\mathbf{X}, \theta)p(\theta|\pi_\theta) \tag{3.2}$$

Which is occasionally written as follows in order to emphasise that the expression is a function of θ:

$$\begin{aligned} \mathcal{L}(\theta|\mathbf{X}, \mathbf{y}, \pi_\theta) &= p(\mathbf{y}|\mathbf{X}, \theta, \pi_\theta) \\ &= p(\mathbf{y}|\mathbf{X}, \theta)p(\theta|\pi_\theta) \end{aligned} \tag{3.3}$$

where $\mathcal{L}$ denotes a likelihood (see appendix A for additional details on this distinction).

We may now formulate a probabilistic model by including explicit expressions for $p(\mathbf{y}|\mathbf{X}, \theta)$ and $p(\theta|\pi_\theta)$. Let:

- $f : \mathbb{R}^d \to \mathbb{R}$ be the function we are interested in creating a model for.
- $\mathbf{X}$ be an $N \times D$ array of N samples from a D-dimensional index.

- $\mathbf{y}$ be a length N vector of sample responses for the function of interest, such that $\mathbf{y}_i = f(\mathbf{x}_i)$ for $i = 1 \dots n$
- $m(\mathbf{X}|\theta)$ be the desired mean function
- $k(\mathbf{X}, \mathbf{X}'|\theta)$ be the desired covariance kernel
- θ be the combined vector of hyperparameters for m and k
- π_θ be the prior distribution for θ

Our probabilistic model is thus:

$$\begin{aligned} \mathbf{y} &\sim \text{MVN}(m_{\mathbf{X}}, K_{\mathbf{XX}}|\theta) \\ \theta &\sim \pi_\theta \end{aligned} \tag{3.4}$$

where MVN is the Multivariate Normal Distribution (MVN; since a GP is an infinite dimensional analogue of the MVN distribution; see appendix B); $m_{\mathbf{X}}$ is a length N vector obtained by evaluating $m(\cdot|\theta)$ at each of the N sample points, $\mathbf{x} \in \mathbf{X}$; and $K_{\mathbf{XX}}$ is an $N \times N$ covariance matrix obtained by evaluating $k(\cdot, \cdot|\theta)$ pairwise for all $\mathbf{x} \in \mathbf{X}$.

The probability of the sample responses, $\mathbf{y}$, given the sample points, $\mathbf{X}$, and hyperparameters, θ, is given by the PDF of the multivariate normal distribution, with mean vector, $m_{\mathbf{X}}$, and covariance matrix $K_{\mathbf{XX}}$, computed as above:

$$p(\mathbf{y}|\mathbf{X}, \theta) = \frac{\exp\left[-\frac{1}{2}(\mathbf{y} - m_{\mathbf{X}})^\intercal \cdot \Omega \cdot (\mathbf{y} - m_{\mathbf{X}})\right]}{\sqrt{(2\pi)^N |K_{\mathbf{XX}}|}} \tag{3.5}$$

where $|K_{\mathbf{XX}}|$ represents the determinant of the covariance matrix, and $\Omega = [K_{\mathbf{XX}}]^{-1}$ represents the inverse of the covariance matrix (sometimes referred to as the *precision matrix*).

The probability of the hyperparameters given their prior is obtained by evaluating the appropriate (joint) probability distribution for θ:

$$p(\theta|\pi_\theta) = \pi_\theta(\theta) \tag{3.6}$$

It is often more convenient to work with the natural logarithm of the likelihood function, since it simplifies obtaining derivatives with respect to θ, and remains a monotonically increasing function:

$$\log(\mathcal{L}(\theta|\mathbf{X}, \mathbf{y}, \pi_\theta)) = \log(p(\mathbf{y}|\mathbf{X}, \theta)) + \log(p(\theta|\pi_\theta)) \tag{3.7}$$

This requires the MVN log PDF:

$$\log(p(\mathbf{y}|\mathbf{X}, \theta)) = -\frac{1}{2}\Big[\log(|K_{\mathbf{XX}}|) + (\mathbf{y} - m_{\mathbf{X}})^\intercal \cdot \Omega \cdot (\mathbf{y} - m_{\mathbf{X}}) + N\log(2\pi)\Big] \tag{3.8}$$

and the log of the prior distribution for θ:

$$\log(p(\theta|\pi)) = \log(\pi_\theta(\theta)) \tag{3.9}$$

3.3 Hyperparameter Inference

We seek the posterior distribution of the hyperparameters, θ; given a (GP) likelihood function which describes the probability of some data, $\mathcal{D}$; and a prior distribution for θ, π_θ. This prior distribution - when discussed in the context of GP hyperparameter inference - is usually referred to as the *hyperprior* to emphasise it is a distribution over model hyperparameters.

The posterior probability of the hyperparameters, $p(\theta|\mathcal{D}, \pi_\theta)$ is obtained by applying Bayes' theorem:

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

where the primary effort resides in computing the marginal likelihood, $p(\mathcal{D}|\theta, \pi_\theta)$. The probability of the data, $p(\mathcal{D})$, is simply a normalisation constant, such that the above relationship can be expressed proportionally:

$$p(\theta|\mathcal{D}, \pi_\theta) \propto p(\mathcal{D}|\theta, \pi_\theta)p(\theta|\pi_\theta)$$

Equivalence between $p(\theta|\mathcal{D}, \pi_\theta)$ and $p(\mathcal{D}|\theta, \pi_\theta)p(\theta|\pi_\theta)$ is only proportional due to the absent denominator, $p(\mathcal{D})$. The product $p(\mathcal{D}|\theta, \pi_\theta)p(\theta|\pi_\theta)$ consequently does not represent a "real" probability distribution as it does not integrate to unity.

We will discuss methods of approximating this distribution with a single point in section 3.4, and more general approximations to the overall shape in section 3.5.

The shape of the posterior is consequently determined by the interaction between the data, the likelihood function (which, in the context of GP regression means the kernel and the mean function), and the hyperprior.

The shape of the posterior marginals for any particular hyperparameter are difficult to determine in advance. For a fixed amount of data, one trades expressive power for identifiability, with the former referring to the number of possible functions the model is capable of representing, and the latter to how uniquely the parameters of that model can be determined given data (roughly, how "dispersed" the posterior marginals are for the model hyperparameters). For GP models, this generally translates to the hyperparameters of kernels capable of modelling more functions being more difficult to identify precisely, all other things remaining equal.

More flexible kernels (i.e., those making fewer assumptions about the target function, and thus invoking a distribution over a larger subset of possible explanations) consequently tend to require more information in order for their hyperparameters to be identified to a given tolerance. Information here refers to the somewhat abstract notion of "something that tells us something" about that hyperparameter. This is typically interpreted to be synonymous with observational data, but also includes the model hyperprior and other sources. Potentially useful examples for engineering applications include the derivatives of observed quantities [19], presence of symmetry or invariances [14, section 2.7], or assumptions of monotonicity [20].

It is also worth noting that the amount of information provided by the data in terms of ability to identify model hyperparameters is non-trivially related to the choice of

kernel and mean function, the locations at which the target function is observed, and the presence or not of any noise. Noisy data, for instance, carries less information than noiseless data.

3.3.1 Model Fit Uncertainty

By understanding hyperparameter identification in terms of statistical inference, we are able to recognise that attempting this procedure implies uncertainty: since "true" values of the hyperparameters for the data generating process cannot be observed from finite data (or, equivalently, do not actually exist), assuming the hyperparameters take particular values introduces the possibility that these hyperparameters are consistent with the observed data but not necessarily data that is out-of-sample. It is worth stressing that this is subtly different to *overfitting*, since it is abstract properties of the data that are encapsulated by the hyperparameters and not the data generating function itself.

As a result, the subset of functions encoded by the GP might not include the "true" data generating mechanism, implying that the predictive distribution cannot be guaranteed to provide either a good approximation to the real data generating process or properly quantify its predictive uncertainty. This is somewhat colloquially referred to as "model fit uncertainty", or uncertainty associated with the inference of model parameters from finite data. This phenomenon is not unique to GP models.

Appreciating that this uncertainty exists and understanding how such inference is performed provides useful insight into how one might try to manipulate the posterior distribution for a particular hyperparameter in order to make it "more identified". Since the posterior is a function of the likelihood (determined by the choice of model), data and hyperprior, a *reduction* in posterior variance could be obtained by using more informative data: implying either more samples, or samples featuring less noise; a less flexible likelihood function, thus increasing the information content of each sample; or a more informative hyperprior, which assigns prior mass across a smaller region of the hyperparameter space.

3.3.2 Hyperpriors

For GP regression, the joint hyperprior is usually defined in terms of independent marginal prior distributions for each hyperparameter.

For example, supposing we define a GP model with hyperparameters $\theta = [\alpha, \rho_1, \ldots, \rho_D]$, to define a joint hyperprior, π_θ, in terms of independent marginals, we need only to come up with prior distributions for α and $\rho_1, \ldots, \rho_D$. The primary benefit of this is that the marginals are much easier to reason with than their joint distribution. A hyperprior which is not explicitly defined for any particular hyperparameter is equivalent to an implicitly defined uniform distribution over that hyperparameter's support: there is thus no loss of generality to assume the existence of the hyperprior for any given hyperparameter (recognising that it is flat if not assumed otherwise).

Uniform hyperparameter priors may have the unintended consequence of distributing large amounts of probability mass over unrealistic values for that variable, particularly for variables which feature unbounded or half-bounded support. Directly relevant examples include the hyperparameters of most popular kernels, such as the amplitude hyperparameter, α, and lengthscale-type parameters, l and ρ, all of which have half-bounded support on $[0, \infty)$. Neglecting to properly consider the priors on such hyperparameters can have consequences on the quality of the resulting model.

A particularly important observation is that in the case of lengthscales, a uniform prior produces a linear bias due to an infinite amount of prior mass allocated beyond the support of the target function. We will observe this in chapter 6.

Effective hyperpriors should also take the parametrisation of the kernel into account (see, for example [21, Ch. 20]). This is noted to be considerably more important when using methods which utilise sampling (such as MCMC), due to potential interactions with the behaviour of the sampling algorithm.

3.4 Point Estimates

Relative to methods which approximate the entire posterior distribution (to be introduced in section 3.5), point estimates of the model hyperparameters are easier to compute and are capable of providing predictions faster due to avoiding integration over the model hyperparameters.

Summarising the posterior distribution with a single point implies the assumption that this point is considered a suitable summary of the entire posterior: this is reasonable if the posterior distribution is "well peaked" and symmetric, though this is a difficult guideline to formalise due to what constitutes a "good summary" being defined in the context of how sensitive important properties of the model are to different hyperparameters.

3.4.1 Likelihood Optimisation

The most common approach to identifying θ is to maximise the log-likelihood, descried by equation (3.8), with respect to θ; i.e. to maximise the probability of the data as a function of the hyperparameters. This is occasionally referred to as a modal approximation [15, ch.13], or a Maximum *A-Posteriori* (MAP) estimate if the prior π_θ is specified explicitly.

Let Θ represent the permissible search space for θ (i.e. a space respecting the constraints on the values that each hyperparameter may take).The Maximum Likelihood Estimate (MLE) for θ, denoted $\theta^*_{\max}$, is given by:

$$\theta^*_{\max} = \arg\max_{\theta \in \Theta} \log(\mathcal{L}(\theta|\mathbf{X}, \mathbf{y}, \pi_\theta)) \tag{3.10}$$

The optimisation described by equation (3.16) is often approached by making use of the fact that the objective function is analytically differentiable. The derivative of the log likelihood with respect to the i^{th} element of θ is given by:

$$\frac{\partial}{\partial \theta_i} \log(\mathcal{L}(\theta|\mathbf{X}, \mathbf{y}, \pi_\theta) = \frac{\partial}{\partial \theta_i} \log(p(\mathbf{y}|\mathbf{X}, \theta)) + \frac{\partial}{\partial \theta_i} \log(p(\theta)) \tag{3.11}$$

where the derivative of the MVN PDF is:

$$\frac{\partial}{\partial \theta_i}\log(p(\mathbf{y}|\mathbf{X},\theta)) = \frac{1}{2}\mathbf{y}^\intercal \cdot \Omega \cdot \frac{\partial K_{\mathbf{XX}}}{\partial \theta_i} \cdot \Omega \cdot \mathbf{y}^\intercal - \frac{1}{2}\mathrm{tr}\left(\Omega \cdot \frac{\partial K_{\mathbf{XX}}}{\partial \theta_i}\right) \tag{3.12}$$

where $\mathrm{tr}(\cdot)$ represents the trace operation and we reiterate that $\Omega = [K_{\mathbf{XX}}]^{-1}$ is the inverse of the covariance matrix.

Application of equation (3.11) to a *specific* problem requires the derivatives of the kernel and the derivatives of the log PDF of the hyperprior with respect to each of the kernel's hyperparameters. Given these derivatives, a bounded Newton or quasi-Newton algorithm can be used to attempt to solve the optimisation described by equation (3.16) to obtain θ^*_{mle}. In practice we find the former to be better behaved, and the reduced memory usage of quasi-Newton methods – such as the popular L-BFGS algorithm of Byrd et al. [22] – is not particularly important for the scale of problems we consider. L-BFGS was also noted to exhibit inconsistent performance in certain circumstances: if initiated from (very) flat regions of the log-likelihood (such that observed changes in the log-likelihood when moving in any direction were of the order of machine precision), then the line-search stage of the algorithm will fail due to inability to identify a suitable search direction. This meant L-BFGS was quite sensitive to where it was initialised in our experiments.

Since θ^*_{max} estimated in this manner is not guaranteed to be a global optimum, multiple restarts and/or a preliminary grid search (to identify an effective start point) are encouraged.

3.4.2 Cross Validation

Rather than direct optimisation, one can also consider choosing the model hyperparameters that maximise the Leave-One-Out Cross-Validated (LOO-CV) predictive density of the data, termed θ^*_{Loo}. Unlike MLE or MAP estimates, cross-validation does not seek to provide a summary of the posterior distribution, but rather attempts to select a point from it which maximises the (perceived) predictive ability of the model. More specifically, LOO-CV seeks to identify the hyperparameters

which maximise the probability of observing each provided data-point, given the others.

The leave-one-out density for the i^{th} element of $\mathbf{X}$ with hyperparameters θ is obtained by constructing a GP using the data minus the left-out point, denoted $\mathcal{D}_{/i} = \{\mathbf{X}_{/i}, \mathbf{y}_{/i}\}$, and calculating the predictive density of this GP at the left out point, $\{\mathbf{x}_i, y_i\}$.

The predictive density is obtained by calculating the probability of the left-out-point, y_i, given the predictive marginal distribution at $\mathbf{x}_i$, which is univariate Gaussian:

$$p(y_i|\mathbf{x}_i, \mathcal{D}_{\text{ı}}, \theta) = \phi(y_i|m^*_{\theta,i}, \sigma^*_{\theta,i}) \tag{3.13}$$

where $\phi(\cdot|\mu, \sigma)$ represents the Gaussian PDF with mean μ and standard deviation σ. The parameters $m^*_{\theta,i}$ and $\sigma^*_{\theta,i}$ are the predictive mean and marginal standard deviation of the GP at $\mathbf{x}_i$ (see equation (B.11) and equation (B.12)), with the subscript θ used to emphasise these are for a given value of the hyperparameters, θ.

LOO scores for a specific value of θ are then obtained by averaging the left-out predictive densities for each of the training points, $i = 1 \dots N$:

$$\text{LooCV}(\theta|\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathcal{D}_{/i}, \theta) \tag{3.14}$$

Computing equation (3.14) as above is noted to be intractably expensive for even moderately large datasets, as it requires computation and subsequent inversion of a total of N covariance matrices of size $N-1$ each time one assesses a hyperparameter sample.

For zero-mean GPs, Sundararajan and Keerthi [23] provide an analytical version which greatly alleviates this computational cost. Let $\omega_\theta = \text{diag}(\Omega_\theta)$ and $\alpha_\theta = \Omega_\theta \cdot (\mathbf{y})$, where $\Omega_\theta = [K_{\theta,XX}]^{-1}$. The LOO-CV score for hyperparameters θ is

then:

$$\text{LooCV}(\theta|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} -0.5\log(2\pi) + 0.5\log(\omega_{\theta,i}) - 0.5(\frac{\alpha_{\theta,i}^2}{\omega_{\theta,i}}) \tag{3.15}$$

Hyperparameters maximising the leave-one-out predictive density can consequently be identified by maximising equation (3.15) over the space of hyperparameters, Θ:

$$\theta^*_{\text{Loo}} = \arg\max_{\theta\in\Theta} \text{LooCV}(\theta|\mathcal{D}) \tag{3.16}$$

using analytical derivatives of equation (3.15), if necessary.

3.5 Approximating The Posterior

Rather than summarising the posterior distribution of the hyperparameters by a single point, it is also possible to construct approximations to the entire posterior. Different methods of accomplishing this construct approximations in different ways, and may be more or less appropriate depending on the specifics of the problem.

An important difference relative to the case of a single value of the hyperparameters is that functions of the GP posterior (such as predictions) must be integrated (or *marginalised*) over the posterior distribution of the hyperparameters if one requires a point estimate of that quantity. Let $p(y^*|\mathbf{x}^*, \mathcal{D}, \theta)$ be the predictive distribution given a specific value of θ. To "marginalise" over the hyperparameters, one integrates over the posterior distribution of θ:

$$p(y^*|\mathbf{x}^*, \mathcal{D}, \pi_\theta) = \int p(y|\mathbf{x}^*, \mathcal{D}, \theta, \pi_\theta)p(\theta|\mathcal{D}, \pi_\theta)d\theta \tag{3.17}$$

The integrated predictive distribution described above is noted to be conditional no longer on θ, rather its prior distribution, π_θ, and the data. Since the integration described by equation (3.18) is not usually analytically tractable, it tends to be

approximated numerically:

$$p(y^*|\mathbf{x}^*, \mathcal{D}, \pi_\theta) \approx \sum_{s=1}^{S} p(y^*|\mathbf{x}^*, \mathcal{D}, \theta_s) w_s \tag{3.18}$$

where the summation is performed over a total of S (optionally) weighted samples of θ, with (possibly equal) weights $w_1, w_2, \ldots w_S$. Functions of the predictive distribution can be marginalised analogously.

3.5.1 Markov-Chain Monte-Carlo

Markov-Chain Monte-Carlo (MCMC) simulation provides a scheme for sampling from an approximation to the posterior distribution of the hyperparameters proportionally to their probability. This yields a series of S equally weighted samples $\theta_1, \theta_2, \ldots \theta_S$ with $w = 1/S$, which asymptotically converge to an unbiased approximation of the true posterior [24].

MCMC is the most expensive but most flexible method of approximating the posterior distribution of the hyperparameters, and is theoretically capable of providing good approximations to arbitrarily complicated posterior densities, including those that are multi-modal or ridge-shaped, if properly configured.

A large number of different MCMC algorithms can be applied to hyperparameter inference for Gaussian Processes. While "classical" MCMC algorithms such as those of Metropolis [25], Hastings [26], and Gibb's sampling [27] exhibit prohibitively slow rates of convergence (equivalently, effective samples per iteration) for all but the most simple GP regression problems, modern algorithms such as slice sampling [28] and Hamiltonian Monte-Carlo [29] offer considerably more efficient sampling behaviour to the extent that modern MCMC algorithms are feasible alternatives to likelihood maximisation or cross validation. In particular, "parameterless" variants such as the No-U-Turn Sampler (NUTS) [30] eliminate the need for manual tuning procedures, which have historically been an obstacle to deploying MCMC algorithms in environments where (expert) guidance necessary for good tuning may be inconvenient to obtain or even unavailable.

3.5.2 Alternatives

Variational Inference (VI) (see [31, 32]) is a family of methods which approximate the posterior distribution by assuming a parametric ("analytical") representation and aiming to minimise the information loss (in a formal sense) between the approximation and the target.

VI is noted to be more scalable than MCMC, potentially scaling Bayesian methods of analysis to much larger datasets than is practical with MCMC: Hensman et al. [33] used a variational re-parametrisation of GP regression which was able to handle several million data points, and Gal et al. [34] showed variational GP regression performed better than many other "highly scalable" models a dataset of 2 million points.

Unlike MCMC, VI methods are generally incapable of producing arbitrarily high-quality representations of the target distribution. Evaluating the quality of this approximation is also not always straightforward, as even in the event that one can be confident that a global minimum in the information loss has been achieved, this value alone is an insufficient measure of the approximation's quality [35].

MCMC is consequently preferable to VI in the situations that it is affordable, though VI methods are emphasised to be a key component in the methodological toolchain required to scale GP regression to larger problems.

3.6 Discussion

Both point estimates and marginalisation over the full posterior distribution of the model hyperparameters have different merits.

Models using point estimates of θ are noted to be up to several orders of magnitude faster to query since they compute a single predictive distribution. However, this distribution is conditional on the choice of θ and thus implies an unquantified source of uncertainty. If the posterior distribution of the hyperparameters is sufficiently peaked, the difference between the marginalised predictive distribution

and the conditional predictive distribution can be small enough such that this uncertainty is negligible. This tends to be assumed implicitly in many engineering focused surrogate modelling applications, but is in general a function of the model, the data, and any prior information.

Using the entire posterior distribution of the hyperparameters in the predictive model implies a more complete treatment of the uncertainty present during the model fit procedure, though we reiterate that obtaining predictions from such models is necessarily slower. This is a *necessary* alternative rather than merely beneficial if any of the posterior marginals of the hyperparameters exhibit a ridge-shaped or multi-modal distribution, since these distributions cannot ever be summarised effectively by a single point.

Point estimates are consequently preferable when either rigorous quantification of predictive uncertainty is not a priority, or very fast feedback from the model is paramount, such as when using a the model to feed an interactive tool for live visualisation. On the other hand, when a model of the uncertainty associated with a surrogate model is required, using the full posterior distribution confers a significantly more complete – and thus robust – representation.

Martin and Simpson [36] reviewed a variant of GP regression as a tool for surrogate modelling, and analysed variants fit using both likelihood maximisation and cross validation for different mean functions. Likelihood maximisation was found to be superior in terms of the *Akaike Information Criterion* [37]. Bachoc [38] also compared point estimates of model hyperparameters for GP regression using MLE to those using CV, finding MLE to be optimal in cases in which the covariance kernel is well specified, reporting consistency with Martin and Simpson [36]. CV was reported to provide more consistent predictive performance in cases of model miss-specification by Bachoc, and the degraded performance of the CV models used by Martin and Simpson was claimed to be due to CV being more susceptible over-parametrisation on higher-dimensional test functions. This is at least intuitively verifiable, as one can imagine maximising the predictive qualities of a more flexible model on a dataset of finite size is more likely to lead to overfitting on that dataset.

In practice, this means that if the properties of the target function are well understood, MLE is the optimal means of representing the distribution of the hyperparameters by a single point. If the properties of the target function are not known, CV may be a more robust alternative, providing the choice of kernel and mean function do not permit over-parametrisation of the problem.

Comparing the performance of fully probabilistic models fit using MCMC to those using point estimates of the hyperparameters is the subject of chapter 6.

3.7 Summary

Fitting a GP to data formally requires identification of the GP kernel's hyperparameters, given that data. Since this data is of finite size, this inference implies a distribution of hyperparameters. This can either be summarised by a single point, or approximated in its entirety, with the former implying the assumption that the selected point is a good representation of the total distribution.

Chapter 4

Loads-Structural Coupling Using Surrogate Models

This chapter documents novel use of surrogate models in an industrial aerospace design environment, providing a practical use-case for the theory introduced in the previous two chapters and an introduction to modelling aircraft loads. At the end of the chapter, we assess an important source of uncertainty in the analysis, which will be a primary motivation for the methods introduced in subsequent chapters.

4.1 Aeroelastic Loads

Motion of an aircraft through the atmosphere implies an imbalance of forces. This imbalance can either be induced intentionally, in which case we refer to a manoeuvre, or due to an atmospheric disturbance, in which case we refer to a gust. Analysis of these forces on the aircraft structure is referred to as an external load. External loads result in internal stresses on the aircraft structure, which must be tolerated for safe operation: put simply, this means that none of the (individual) components which compromise the structure should be subject to stresses which could result in their permanent deformation during operation. The structure must consequently be designed with such stresses in mind, and to accomplish this, the loads which produce them should be understood. It follows that calculation of external loads is one of the most important phases of the aircraft design cycle, and performing these calculations quickly and accurately can potentially provide benefits in terms of either enhanced design space exploration or reduced design program time (and consequently, reduced program cost).

Physically, external loads are composed of three continuously distributed forces (acting in each of the three orthogonal 3D-Cartesian axes) and their corresponding moments, totalling six possible "degrees of freedom" (DOFs). For the purposes of analysis, these continuous measurements tend to be discretised, such that one expresses external loads as vector, $\mathbf{y}$, which consists of 6 DOFs measured at a total of M discrete locations or "nodes". Each of the (total of) $6M$ individual elements of $\mathbf{y}$ is referred to as an Interesting Quantity (IQ), and constitutes a particular DOF measured at a particular node. Calculation of these IQs is synonymous with calculation of the external loads.

In many cases, these loads are *transient*, such that $\mathbf{y}$ is a function of time. Not only may the mechanism by which the forces are generated feature time-varying properties (which can be easily imagined in the case of, for example, turbulence), but since the structure of an aircraft is flexible, deformation of the aerodynamic surfaces under load can produce a change in the distribution of pressure and consequently the aerodynamic loads themselves. The activation of load-alleviation systems may

also be an important consideration. Time-varying loads are denoted $\mathbf{y}(t)$.

Exactly how the aerodynamic surfaces of an aircraft deform under load is a function of the structure and the type of load, such that $\mathbf{y}(t)$ can be (somewhat abstractly) expressed as a function, f, which accepts two vectors: $\mathbf{z}$, which represents a parametrisation of the structure; and $\mathbf{x}$, which represents a parametrisation of the loading "case". A load case encapsulates a description of the conditions the aircraft is subject to which are responsible for producing the load, and will be elaborated on shortly.

A loads calculation can hence be abstracted as follows:

$$\mathbf{y}(t) = f(\mathbf{z}, \mathbf{x}) \tag{4.1}$$

4.1.1 Loads Process

The function, f, is a black-box which we will refer to as the *loads process*, and calculates the loads acting on the aircraft as a function of a given load case and aircraft design. Representing the loads process as a single function is stressed to be an abstraction of the real process, which is better described as an "ecosystem" of different uni-disciplinary models such as CFD and both steady state and time-domain structural solvers. We will, however, refer to the loads process as a single function or model, as we are not concerned with precisely what goes on inside it, insofar as this means "opening up" any of its constituent uni-disciplinary components.

In this document, we consider a certification-standard loads model of the full aircraft, which incorporates a complete representation of the aircraft's structure and control surfaces. This is an expensive and time consuming model to query, but provides a high degree of accuracy in the load calculations.

By default, f is assumed to return the time history of the loads, $\mathbf{y}(t)$. These quantities are sometimes referred to as *correlated* loads as the evolution of each DOF at each node through time is returned, hence preserving the relationships between different IQs. For the purposes of a detailed structural analysis, the correlated loads

are essential, as internal stresses are a function of all the IQs simultaneously: letting $\mathbf{t}$ be the vector of internal forces acting on the structure, one obtains $\mathbf{t}$ via another function, g, of $\mathbf{y}(t)$:

$$\mathbf{t} = g(\mathbf{y}(t)) \tag{4.2}$$

It is $\mathbf{t}$ that is of interest to structural analysts.

In practice, the function g is expensive and cumbersome to work with. In this work we consider an approximation to the correlated loads, such that $\mathbf{y}(t)$ is approximated by a single vector $\mathbf{y}^\dagger$. This approximation is as simple as taking the maximum absolute value of each of the elements of $\mathbf{y}(t)$ over the duration of the load, noting that the time at which the maximum of a particular IQ occurs may be different for another:

$$\mathbf{y}_i^\dagger = \max_t \ |\mathbf{y}_i(t)| \tag{4.3}$$

where the subscript i represents the i^{th} element of the subscripted vector. We summarise this operation by a function, h, such that $\mathbf{y}^\dagger = h(\mathbf{y}(t))$.

Replacing $\mathbf{t}$ with $\mathbf{y}^\dagger$ in this manner does not represent a truly realistic design scenario, as the two quantities are not in any way equivalent. However, we assert that this is sufficient for the experiments conducted in this work: the approximation described by h is simply a black-box that could just as easily be replaced with the real black-box g in a production environment, and we will treat the elements of $\mathbf{y}^\dagger$ analogously to how we would those of $\mathbf{t}$.

For notational simplicity we will represent the loads process, f, and subsequent condensation of $\mathbf{y}(t)$ into $\mathbf{y}^\dagger$ via h as a single function, l, and consider it implicit that $\mathbf{y} \equiv \mathbf{y}^\dagger$ in the absence of a time argument, such that the loads process is

represented as follows:

$$\mathbf{y} = l(\mathbf{x}, \mathbf{z}) \equiv h(f(\mathbf{x}, \mathbf{z})) \tag{4.4}$$

4.1.2 Load Cases

A single load case, described by a vector, $\mathbf{x}$, describes a situation in which forces are applied to the aircraft, either as a result of pilot-induced motion or a temporary disturbance in the atmosphere. This vector may include discrete, conditional and continuous properties which together define a single load case. We offer a (non-exhaustive) list of such parameters.

Discrete parameters: case type (*Gust/Manoeuvre*), presence of control laws, control surface configurations.

Continuous parameters: Mach number, altitude, aircraft mass, thrust level, centre of gravity position, fuel fill levels.

Conditional parameters: gust velocity (*conditional on case type*), manoeuvre sub-type (*conditional on case type*), turn G-force (*conditional on manoeuvre type and subtype*).

The space of *all* load cases will be denoted $\mathcal{X}$, and a single load case by $\mathbf{x}$.

Adopting this notation, a single load case defined in terms of case type, Mach number, altitude, aircraft total mass, and centre of gravity position could be described as follows:

$$\mathbf{x} = [\text{Type}, \text{Mach}, \text{Altitude}, \text{Mass}, \text{CG}]$$

We make note that for the purposes of this document this is restricted specifically to flight loads: ground loads are not considered here.

4.1.3 Design Vectors

The design vector, $\mathbf{z}$, contains a parametrised description of the aircraft. In principle, this contains a complete aircraft design, though in practice only a subset of the total number of potential design variables tend to be of interest at any given time. As a simple example, adaptation of wing structural definition to accommodate a new type of engine may not need to reconsider design of the fuselage.

In this manner, one can consider *derivative* designs, in which the majority of the design is fixed, and only a (relatively) small number of parameters are considered to be "active" design variables. This is necessary, as the techniques developed in this document do not scale to the huge number of design variables required for complete parametrisation of a full aircraft design. This document will consider only the wing, with the significance of individual design variables clarified when necessary. Analysis of different types of derivative designs using the techniques we develop here is possible, though we stress again that the methods do not scale to conceptual and very-early design phase situations in which either a much larger number of variables can be considered free, or a suitable method of parametrisation may not be obvious.

The space of all (derivative) designs currently under consideration (noted again to be much smaller than the space of all designs) will be denoted $\mathcal{Z}$, and a single design by $\mathbf{z}$.

4.2 Loads Structural Coupling

Aeroelastic modelling of flight loads is an important stage of aircraft design, as it is a necessary pre-requisite to performing structural optimisation – a procedure that is critical to meeting performance standards.

For the purposes of this work, such constraints are designated via a function of the maximum absolute value for each IQ; that is, the structure is designed to a tolerance

determined by the maximum loads it encounters across all possible load cases:

$$\overline{\mathbf{y}} = \max_{\mathbf{x} \in \mathcal{X}} |l(\mathbf{x}, \mathbf{z})| \tag{4.5}$$

where the max operator operates componentwise for each element of the vector $\mathbf{y} = l(\mathbf{x}, \mathbf{z})$ for all load cases $\mathbf{x} \in \mathcal{X}$. The resulting vector of maximum absolute loads, $\overline{\mathbf{y}}$, is referred to as the *envelope*. This is again stressed to be an approximation, as in a production environment it is the maximum internal stresses, $\mathbf{t}$, that constrain the structure, though this approximation also receives industrial use as a useful approximation.

With reference to equation (4.5), identification of $\overline{\mathbf{y}}$ requires the design, $\mathbf{z}$, to be designated: this is a problem, as structural definition requires the envelope, $\overline{\mathbf{y}}$. In other words, the design process necessitates a feedback loop due to the coupling between the envelope and the structural definition of the aircraft.

This feedback loop, sometimes referred to as a *loads cycle*, presents a difficulty, as both querying the loads process and (particularly) structural optimisation are computationally intensive procedures. Designers consequently have two options:

1. Iterating through the feedback loop until structural design convergence. This is explored by Iorga et al. [2]: after each external loads calculation, the aircraft structure is re-optimised in terms of mass while satisfying structural requirements calculated using the external loads calculated for the design obtained at the end of the previous cycle (or the baseline design in the case of the first iteration). The external loads are then re-calculated for the optimised structure, and the process repeated until the optimised structure does not change (within some tolerance) between iterations.

2. Accepting design uncertainty and working with uncertain loads due to uncertainty in the structural definition of the design. This necessitates a realistic model of such design uncertainties, which is challenging. A facet of this problem is explored by Krupa et al. [3]: by constructing a surrogate model of the loads as a function of derivative structural changes, the authors are able to

propagate uncertain design vectors through the loads process using Monte-Carlo simulation, modelling dependency between the constituent design variables using a copula.

In the first case, use of a surrogate model for the loads process can significantly alleviate associated computational costs, and in the second it is essential in order to feasibly perform the sample based simulation required to propagate correlated uncertainties. We hence seek a model which is capable of approximating the envelope, $\overline{\mathbf{y}}$, for an arbitrary derivative design, $\mathbf{z} \in \mathcal{Z}$.

Computing the envelope requires identification of the load cases at which the absolute maximum values of each IQ occur. This subset of cases is referred to as the *critical load cases*.

4.3 Critical Load Cases

An arbitrary load case, $\mathbf{x}^*$, is critical if:

$$|l(\mathbf{x}^*, \mathbf{z})| \geq |l(\mathbf{x}, \mathbf{z})| \quad \forall \mathbf{x} \in \mathcal{X} \tag{4.6}$$

holds for at least one element of $\mathbf{y} = l(\mathbf{x}, \mathbf{z})$. The *set* of critical load cases, $\mathcal{X}_c$ contains the critical load cases for all quantities of interest, such that the envelope, $\overline{\mathbf{y}}$, is obtained by:

$$\overline{\mathbf{y}} = \max_{\mathbf{x} \in \mathcal{X}_c} |l(\mathbf{x}, \mathbf{z})| \tag{4.7}$$

Identification of the envelope, $\overline{\mathbf{y}}$, is thus synonymous with identifying the set of critical cases, $\mathcal{X}_c$.

4.3.1 Case Downselection

With reference to equation (7.2), it is clear that since the loads process, $l(\mathbf{x}, \mathbf{z})$, is a function of the design, $\mathbf{z}$, that the set of critical cases must be a function of that

particular design, too. Identification of $\mathcal{X}_c$ outright implies performing the following optimisation:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} |l(\mathbf{x}, \mathbf{z})| \tag{4.8}$$

for *each* IQ.

This is intractably expensive: not only is the loads process an expensive function, but the space of all load cases, $\mathcal{X}$, is large and challenging to navigate; Cooper et al. [39] put the number of individual load cases at around 10 million assuming discretisation of the continuous parameters into 50 flight points.

However, $\mathcal{X}_c$ can often be reliably estimated using information from previous analyses. This can take the form of either data from past studies, recommendations from domain experts, or a combination of the two. Given the set of critical cases, one can then consider building a model of the envelope as a function of $\mathbf{z}$.

4.4 Data, Notation, and Problem Description

Our task is to provide rapid feedback of changes to the envelope, $\overline{\mathbf{y}}$, as a function of potential design changes: We hence seek a model of the envelope as a function of design changes under consideration $\mathbf{z} \in \mathcal{Z}$.

The design changes in these experiments correspond to 10 wing design parameters: 2 geometric twist variables (inner and outer), 4 bending stiffnesses (inner, mid, outer, winglet), and 4 torsional stiffnesses (inner, mid, outer, winglet). The design variables are scaled such that the design space is a 10-dimensional hypercube hence $\mathbf{z} \in [0, 1]^{10}$.

To build a model of $\overline{\mathbf{y}}$ as a function of $\mathbf{z}$, a set of K critical load cases $\mathcal{X}_c \triangleq \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}$ is presumed to have been identified a-priori for the IQs under consideration.

Conceptually, these load cases are then "partially applied" to $l(\mathbf{x}, \mathbf{z})$ such that the

loads analysis process is rendered discrete over $\mathcal{X}$:

$$l_k(\mathbf{z}) \triangleq l(\mathbf{x}_k, \mathbf{z}), \quad k = 1, \ldots, K \tag{4.9}$$

i.e., rather than a single function of both $\mathbf{x}$ and $\mathbf{z}$, we consider K individual functions of just $\mathbf{z}$, corresponding to each of the critical load cases.

A model of the envelope hence involves constructing K individual models of $\mathbf{y}$ as a function of $\mathbf{z}$ for specific values of $\mathbf{x}$. This simplifies the problem considerably, as we may consider K individual, independent approximations for l_k, substantially alleviating the curse of dimensionality and obviating the need to explicitly model any coupling between $\mathbf{x}$ and $\mathbf{z}$, which is a challenge.

This approach to constructing surrogate models of loads is used in industry focused studies [2] and by the majority of academic texts concerning models for flight loads.

4.4.1 Design of Experiments

For the purposes of our experiments, we are granted a (slightly) arbitrary budget of 200 * K runs of the loads process, corresponding to investigation of 200 different design vectors at each of the pre-identified critical load cases. Though this is a relatively large number of simulations, it is of a similar order of magnitude to the quantities of data currently used in typical design programs for derivative designs, though it should be noted that this does depend on the specifics of the design program.

We use the min-max Latin-Hypercube Sample (LHS) design of Morris and Mitchell [40] to draw $N_{\text{total}} = 200$ samples from the $d = 10$ dimensional vector design space, then randomly permute and partition these into a 3:2 ratio train-test split, resulting in $N = 120$ and $N_{\text{test}} = 80$. Since the purpose of our experiments is to investigate the performance of models constructed using this acquired data, a generous test split is used to ensure the test points provide good coverage across the 10-dimensional space. A more typical train-test split of 8:2 featuring a less generous test split would likely

produce more accurate models due to increased "information" being available during model fitting, though could have potentially compromised our ability to assess these models with confidence. Likewise, a 1:1 split was seen as too unrepresentative of a production environment in which the test split tends to be squeezed as much as possible in order to provide the most amount of data possible for model fitting.

Let $\mathbf{Z}_{\text{train}} \triangleq \{\mathbf{z}_1, \dots \mathbf{z}_N\}$ be the set of 120 training points generated via min-max LHS. We evaluate $l_k(\mathbf{z})$ for the same set of sample points, $\mathbf{Z}_{\text{train}}$, for each of the K load case functions.

Since each l_k returns a vector, $\mathbf{y}_k$, for given $\mathbf{z}$, we obtain a matrix of responses $\mathbf{Y}_k$ for each of the K load case functions. For an arbitrary dataset consisting of N samples:

$$\mathbf{Y}_k \triangleq \begin{bmatrix} \mathbf{y}_{k,1} \\ \vdots \\ \mathbf{y}_{K,N} \end{bmatrix} \triangleq \begin{bmatrix} l_k(\mathbf{z}_1) \\ \vdots \\ l_k(\mathbf{z}_N) \end{bmatrix} \tag{4.10}$$

$\mathbf{Y}_k$ has M columns, equal to the number of IQs under consideration, and N rows, corresponding to the number of sample points.

From here we omit the subscript k when it is implicit that the analysis is performed separately for each load case function, except where necessary for clarity.

4.4.2 Calculating the Envelope

Given a total of M IQs, the envelope is a length M vector, $\overline{\mathbf{y}}$, defined for each design as the maximum absolute value for each IQ across all load cases.

Following the notation introduced previously, and assuming we investigate a total of N different derivative designs, each load case is assumed to return an $N \times M$ array of loads, $\mathbf{Y}$.

To compute the envelope, $\mathbf{Y}$ is computed for each of the K load cases, and the results aggregated into an $N \times M \times K$ array, $\widehat{\mathbf{Y}}_{nmk}$. The envelope for each of the

N designs is obtained by taking the coefficient-wise absolute maximum across the third axis of $\widehat{\mathbf{Y}}_{nmk}$:

$$\overline{\mathbf{Y}} = \max_{k=1\ldots K} |\widehat{\mathbf{Y}}_{nmk}| \tag{4.11}$$

such that $\overline{\mathbf{Y}}$ is equivalent to the N vertically stacked envelopes for each of the considered designs:

$$\begin{bmatrix} \overline{\mathbf{y}}_1 \\ \vdots \\ \overline{\mathbf{y}}_N \end{bmatrix} \equiv \overline{\mathbf{Y}} \tag{4.12}$$

4.5 Preprocessing: Linear Dimension Reduction

A naiv̈e approach to modelling $\mathbf{y}$ as a function of $\mathbf{z}$ is to treat each of $\mathbf{y}$'s M scalar elements as independent, and construct a separate model for each of them. This is inefficient and cumbersome, as M is typically large and the correlation information present in $\mathbf{y}$ is discarded. The latter is especially problematic for downstream processes, which compute quantities as functions of $\mathbf{y}$, since assumed independence can produce non-physical results if the mechanism by which $\mathbf{y}$'s elements are coupled is not properly respected (for instance, division of very small numbers by one another).

To remedy this, we apply linear dimension reduction by means of Principal Component Analysis (PCA; equivalently referred to as *matrix factorisation via SVD* or "POD" for Proper Orthogonal Decomposition) on the data matrix $\mathbf{Y}$. This permits regression on an *independent* basis of significantly fewer variables and ensures linear coupling between outputs is preserved. For an accessible introduction to PCA in the context of dimension reduction, we direct the reader to [41]. This procedure is described in detail for our problem in appendix D.1.

Suppose we multiply $\mathbf{Y}$ by a matrix $\mathbf{P}$ obtained via PCA to obtain a low rank approximation $\mathbf{Q} = \mathbf{YP}$. The columns of $\mathbf{Q}$ correspond to $R < M$ variables in a re-

duced dimensional basis, which are (by definition) computed as linear combinations of the original M variables. We assert that a strictly linear change of basis does not compromise the method as significantly non-linear correlations between IQs are not anticipated. This approach has been applied in the context of modelling correlated loads in similar academic research, [42], and is also deployed in the industrial case study of Iorga et al. [2].

Once again we stress that this is performed separately for each load case. This is because the correlation structure between IQs is not anticipated to be equivalent under different load cases: in other words, we expect that the way in which different force measurements are correlated with one another as a function of small changes to the design will be a function of the type of loading that is applied. Tensor factorisation which *includes* the load case axis was explored by Tartaruga et al. [43].

4.5.1 Selecting the Number of Basis Terms

An important decision when deploying PCA for the purposes of dimension reduction is deciding an appropriate value for R, the number of compressed dimensions. By definition, as $R \to M$, one will recover all of the observed variation in the original data, however this defeats the purposes of PCA as a tool for dimension reduction. We select R via assessing the residual error on the resulting envelope associated with compression and subsequent recompression from R dimensions. We choose $R = 10$. This is documented in appendix D.3.

Finally, though the number of compressed dimensions, R, could be tailored to each individual load case, we select R for all of them at the same time in the interest of simplicity.

4.6 Regression Model

We perform GP regression on each of the R columns of the compressed training data matrix, $\mathbf{Q}_{\text{train}} = \mathbf{Y}_{\text{train}}\mathbf{P}$, with respect to $\mathbf{Z}_{\text{train}}$, where $\mathbf{P}$ is a matrix of r PCA

basis vectors.

Let a single column of $\mathbf{Q}_{\text{train}}$ be a represented by a length N vector $\mathbf{q}.$, i.e:

$$\mathbf{Q}_{\text{train}} = \begin{bmatrix} \mathbf{q}_1 & \dots & \mathbf{q}_R \end{bmatrix} \tag{4.13}$$

The R regression targets, $\mathbf{q}_1 \dots \mathbf{q}_R$, are (linearly) independent by construction due to the projection onto the PCA basis, and may thus be modelled independently.

4.6.1 Assumptions and Fit Procedure

Though it is intuitively difficult to interpret how the compressed loads, $\mathbf{q}$, should behave with respect to changes in the design, $\mathbf{z}$, a number of general observations are possible without knowledge of the physics of specific load cases.

Since the range of design changes under consideration are perturbations about a baseline design, one can expect the resulting loads to experience locally linear variation with respect to the design variables under consideration. Since PCA is a linear transformation, this implies that the behaviour of the compressed loads should be approximately linear as a result.

We thus anticipate that the behaviour of $\mathbf{q}$ as a function of $\mathbf{z}$ ought to be at least smooth over $\mathcal{Z}$, and consider the EQ kernel as a reasonable choice. As we do not expect the function to be complex and we have a sufficient density of points such that long range predictions are not required, a zero-mean function is a simple and effective choice, ensuring that all of the variation in the data is modelled by the GP without risking over-parametrisation.

We fit the model using a maximum likelihood estimate via a multi-restart Newton method. This is preferable to MCMC in this instance for three reasons:

1. There is a sufficient volume of training data such that point estimates of the posterior distribution of the hyperparameters are effective summaries. This can be validated by assessment of θ's posterior distribution for different components (i.e. columns of $\mathbf{Q}$) and load cases.

2. The procedure must be repeated RK times, thus inflating the difference in clock-time between MCMC and MLE. This is less of a problem in production (where only a single instance of each model needs to be trained), and is more of an issue when prototyping different models.

3. The intended application does not make use of the posterior variance: the models are used as response surfaces to produce point predictions only.

We thus fit a total of RK GP models: one model for each of the R independent values in the compressed loads space, for all K load cases.

4.6.2 Point Prediction

A terse description of the process by which point predictions of the envelope for a new design, $\mathbf{z}^*$, is described by algorithm 1. A detailed description is documented in appendix D.4.

Algorithm 1 Obtaining an envelope prediction for a new design vector

Given: new design vector, $\mathbf{z}^*$; K sets of R GP emulators for each component of the compressed loads for each load case, $\mathcal{GP}(\cdot|\cdot, \mathcal{D}, \theta)$; K PCA matrices, $\mathbf{P}$; K sets of R rescaling functions, $\phi^{-1}(\cdot)$.
for $k = 1, \ldots, K$ **do**
    for $r = 1, \ldots, R$ **do**
        $\hat{q}^*_{kr} = \phi^{-1}_{kr}(\mathbb{E}[p(q^*_{kr}|\mathbf{z}^*, \mathcal{D}_{kr}, \theta_{kr})])$ ▷ GP rescaled predictive mean.
    end for
    $\hat{\mathbf{q}}^*_k = [\hat{q}^*_{k1}, \ldots \hat{q}^*_{kR}]$
    $\hat{\mathbf{y}}^*_k = [\hat{\mathbf{q}}^*_k \cdot \mathbf{P}_k]^\intercal$
end for
Vertically stack the row vectors $[\hat{\mathbf{y}}^*_1 \ldots \hat{\mathbf{y}}^*_K]$ to obtain $K \times M$ array $\widehat{\mathbf{Y}}^*$.
Compute the absolute maximum of each column of $\widehat{\mathbf{Y}}$ to obtain $\overline{\mathbf{y}}^*$
Return: $\overline{\mathbf{y}}^*$

4.6.3 Predictive Uncertainty

A quantification of the uncertainty associated with a prediction can be obtained by assessing the marginal variance of the predictive distribution at a particular

point:

$$\sigma^2(q^*) = \mathbb{V}[p(q^*|\mathbf{z}^*, \mathcal{D}, \theta)] \tag{4.14}$$

This predictive uncertainty, however, occurs in the compressed space. To understand the influence of this uncertainty in the loads space, the uncertainty in the compressed loads must be propagated through algorithm 1. However, in the present application, the marginal predictive variances are sufficiently small (of the order $1e^{-6}$, after rescaling) such that any uncertainty in the resulting loads is significantly smaller than that due to the PCA basis. We are thus able to ignore the effect of predictive uncertainty in predicting the compressed loads, since the decompression error associated with use of dimension reduction via PCA is several orders of magnitude larger.

4.7 Results

Correlation plots of (normalised) predicted enveloping loads against actual enveloping loads for the test partition are shown in figure (4.1). The linear correlation coefficients for the three DOFs are all close to 1 (closer than $1e^{-3}$ in the case of the shear and moment measurements). An anomalously poor prediction (in terms of *absolute* error) can be observed for a single shear measurement. In general predictions for the torsional envelope appear to exhibit the greatest residual error. Mean absolute PCA reconstruction errorbars are present on the plot, but are not visible due to being very small. Good agreement between the PCA-assisted surrogate models and the loads-process generated data indicates the assumption of quasi-linearity in the external loads (and their linearly transformed compressed representation) with respect to the small, derivative design changes for the variables considered here is sound.

Figure (4.2) shows the distribution of relative errors $\varepsilon_{\text{relative}} = (\overline{y}_{\text{pred}} - \overline{y}_{\text{test}})/\overline{y}_{\text{test}}$ for each of the shear, moment and torque measurements. We report the log cumulative empirical density of the relative residual errors, taking the logarithm of the density

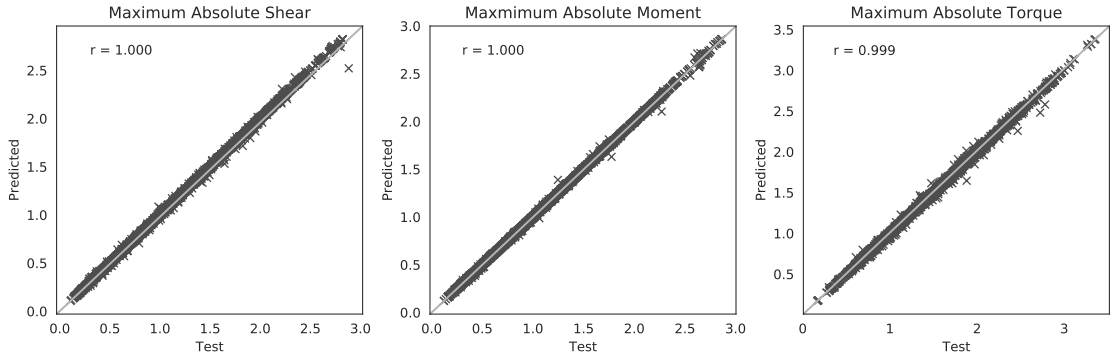

Figure 4.1: Correlation plots for the predicted and test enveloping loads. Errorbars due to PCA reconstruction error are shown but are small (less than $1e^{-2}$).

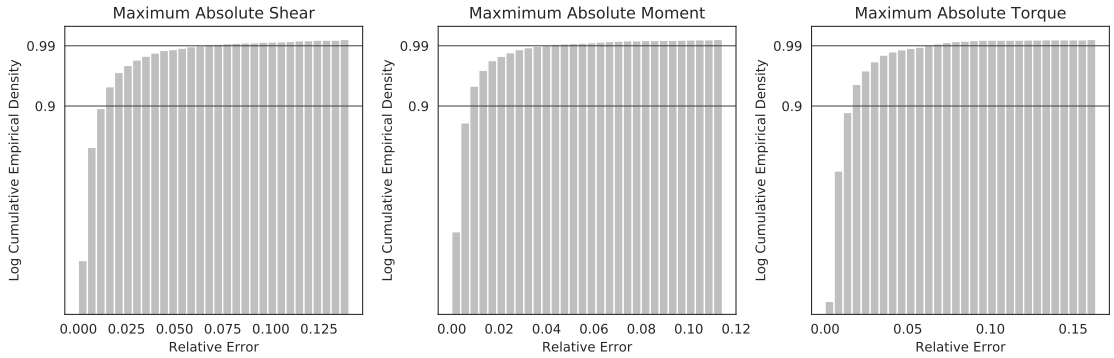

Figure 4.2: Distribution of error for the test predictions. The solid lines represent cumulative densities of 0.9 and 0.99.

in order to resolve the presence of outliers. The two solid lines across the x-axis represent cumulative densities of 0.9 and 0.99, in ascending order, such that the point at which the histogram crosses the line indicates the total fraction of the data with a relative error less than that value. In all cases, around 90% of the data has a relative error of less than 2.5%, and around 99% of the data has a relative error of less than approximately 5% for the shear and bending measurements. For torsion, 99% of the data has a relative residual error of less than approximately 7%. The highest observed relative error is of the order of 15%.

4.8 Analysis

The surrogate modelling technique described above is capable of producing high quality approximations of the enveloping loads as a function of small changes to

parametrised wing stiffness and geometric twist design variables, assuming the critical cases have been appropriate identified. Similar models were deployed directly in a production environment in [2], and used to propagate correlated uncertainties in wing stiffness variables in Krupa et al. [3].

Almost all the error in this analysis is due to use of a compressed representation of the loads and associated loss in fidelity. Use of more basis terms in the compressed representation could resolve this issue at the expense of a more expensive model construction procedure. In a production environment, this is reasonably inconsequential, as the time taken to generate the data in the first place is several orders of magnitude larger. We could thus anticipate increasing R to produce better quality approximations if deploying such models to production.

4.9 Discussion

In this application, a set of critical load cases was presumed to have been identified a-priori for the IQs under consideration. This is a strong assumption, as a direct search for the critical cases is never actually performed for any specific design that is not the baseline configuration.

In practice, this works well for designs which are derivatives about a baseline, as the critical load cases are not expected to change significantly, and the estimated set of critical cases can be assumed "close enough" to the actual critical cases. There is, however, still uncertainty associated with this assumption, as the optimisation to find the true critical cases is never actually performed.

A more serious issue is encountered when considering novel designs or more radical design reconfigurations. In these situations, expert recommendations may be unreliable or vague, and large amounts of quantitative data will not exist. There consequently exists no rationale to performing case downselection reliably without first studying the load case space: this is an expensive and time consuming endeavour due to the size of the load case space, and can contribute significantly to design program time and monetary expenditure on computational power.

We will tackle the problem of critical load case identification in chapter 7, and modelling uncertain loads due to design variability in chapter 8. We will discuss how to perform qualitative load case downselection using data, along with methods for acquiring this information in a sample efficient manner. Since both of these techniques require high quality representations of model uncertainty to function reliably, methods to construct robust models of predictive uncertainty will consequently be the subject of the next two chapters as we develop necessary modelling tools to tackle the industrial problem.

4.10 Summary

We have introduced the problem of loads structural coupling, along with the aircraft loads process, and the concept of a critical load case. We studied a method for constructing surrogate models of the enveloping loads as a function of small, simplified design changes, under the assumption that the conditions at which these critical loads occur are known. High agreement was observed with validation data. Finally, we discussed the consequences of relaxing the assumption that the critical cases are known, and identified situations in which this is a relevant concern, motivating development of robust models of uncertainty to be introduced in the following chapter.

Chapter 5

Robust GP Emulation

In this chapter we discuss the construction of robust models of uncertainty which we term *robust emulators*. We discuss some of the obstacles to fitting fully probabilistic GP models in an industrial environment, and provide carefully considered techniques for navigating them.

5.1 Robustness for Surrogate Models

Surrogate models are deployed for a variety and increasingly diverse assortment of different engineering purposes. For many of these, quantification of predictive uncertainty is beneficial, while for others it is essential: If replacing a physics based model with a surrogate, downstream analyses that are sensitive to the model outputs may wish to assess the robustness of a given decision given potential uncertainty associated with the use of a surrogate. Surrogate models used for uncertainty propagation must account for their own predictive uncertainty in the resulting posterior distribution.

An occasionally overlooked source of potential uncertainty associated with the use of a surrogate model, particularly in industrial contexts, is the uncertainty associated with inference of the model (hyper)parameters. Many surrogate models deployed in production environments utilise point estimates, which – with reference to chapter 3 – assume that the distribution of the parameters given the data can be represented by a single point. While this assumption is often satisfied, it is by no means a guarantee, and if rigorous quantification of uncertainty is required, an important source of uncertainty might be overlooked as a result. It is noted that this is not unique to GP regression.

It follows that utilising the full distribution of the model parameters (occasionally termed a "fully probabilistic" treatment) is attractive, though faces implementation challenges in practice: it is not always clear how to construct effective prior distributions for the model parameters, and computing the posterior distribution appropriately has historically been considered to be a temperamental, computationally expensive task. Flexible methods capable of providing good quality approximations to the posterior distribution, such as Markov-Chain Monte-Carlo (MCMC), are significantly more expensive than optimisation, require the model to be expressed probabilistically, and (in the case of MCMC especially) might require time spent tuning the sampling algorithm. Industry has historically been reluctant to engage with such compromises except when absolutely necessary.

This chapter introduces the theoretical conditions for robust GP regression [44]. We will explore construction of generally applicable prior distributions that are effective for most practical GP regression problems, and show that modern MCMC algorithms can provide fast and effective "turnkey" sampling of model hyperparameters. The next chapter then demonstrates these benefits.

5.2 Defining a Robust Emulator

Robustness from the perspective of a model implies it can be applied to a wide range of potential problems with consistent performance. In the context of emulation specifically, this means that the model should consistently provide good quality statistical predictions, including quantification of predictive uncertainty.

There are also practical facets to robustness. The method should be transparent in its assumptions, provide clear diagnostics in the cases it is unsuitable, and be straightforward to apply and assess. Methods which do not adhere to this are unlikely to be useful in practice.

5.3 Model Choice

As introduced in chapter 2, GP regression provides a fully probabilistic model of supplied data conditional on a number of assumptions about the process by which that data may have been generated. Since the data generating mechanism in surrogate modelling applications is usually a simulator (a computer model of a physical phenomenon), "properties" of this function can often either be extracted from knowledge about the underlying physical process, or – at the very least – evaluated against expert understanding of the problem to assess their suitability.

Selecting a covariance kernel ought only to require context specific considerations. Given the appropriate scripting tools, a GP prior can be "designed" by matching various context-specific assumptions from a menu of available options to the appropriate covariance kernel as described in chapter 2 and appendix C. Such models can be realised by using relatively simple metaprogramming techniques: that is,

by having a higher level program or scripting language seamlessly connect different low-level code "blocks" representing different choices of prior distributions and covariance kernels together, to then be compiled in order to provide efficient run-time performance.

In the following sections, we consider a stationary, anisotropic GP with hyperparameters $\theta \triangleq [\alpha, \rho_1, \ldots, \rho_d]$, where α is the marginal standard deviation and $\rho_1 \ldots \rho_D$ are D anisotropic lengthscale parameters. This covers the majority of GP regression problems as encountered in surrogate modelling applications.

5.4 Generally Applicable Hyperpriors

It is often the case that only a limited amount of information is known about the properties of the target function before the model is fit to data. Nevertheless, even such limited information is useful and helps avoid use of flat (or equivalently, undefined) prior distributions. We refer to these considerations as "generally applicable" for the reason that they can be used in the absence of problem specific information.

It should be stressed that "informative" hyperpriors produced by expert elicitation and/or past studies are almost always preferable: these can significantly reduce the effects of model fit uncertainty, though their availability is a relative luxury. While we note and emphasise that synthesis of such information is possible using this framework (one simply adjusts the prior distribution for the relevant hyperparameter(s)), this is not the focus of this section.

5.4.1 Scaling

An important pre-requisite to defining generic prior distributions is appropriate scaling of the observed data. Both the input and output data should be scaled appropriately, since this assists in "scale-free" treatment of the model hyperparameters, which is significantly more generalisable.

Model inputs should be transformed onto the unit scale, such that the support

of the target function is $[0,1]^D$ (with D being the number of input dimensions). Similarly, model outputs should be scaled such that they have approximately zero mean and unit standard deviation. This is easily accomplished by subtracting the sample mean and dividing by a robust measure of the dispersion, such as the Median Absolute Deviation (MAD) [21]. Using the sample standard deviation is not advised as it is sensitive to outliers, and obtaining a standard deviation of precisely 1 is not necessary.

5.4.2 Priors for Marginal Standard Deviation

Ensuring the marginal standard deviation, α, is not too high reduces the tendency of a GP model to "overfit" the data with short lengthscale functions. Given the output data is pre-processed to have approximately unit standard deviation, we can encode the belief that the marginal standard deviation is of approximately unit magnitude by using a half standard-normal prior distribution centred at 0, following recommendations in the Stan user manual [21]:

$$\alpha \sim \text{Normal}^+(0,1) \tag{5.1}$$

where Normal^+ refers to a (positive) half-normal distribution. In the context of a GP likelihood, when compared to a flat or thick-tailed prior distribution this choice is "weakly informative" in the sense that it constrains α to around approximately unit magnitude, but otherwise lets the likelihood (the other component of the posterior) "speak for itself".

This choice is noted to be somewhat arbitrary, and one could also consider (for example) a half student-t $(4,0,1)$ distribution instead. More informative distributions could be used for specific applications these though are significantly less flexible.

Experimentation with the marginal standard deviation prior may produce (more) effective bespoke options for specific use cases, though this prior distribution is significantly less important than that of the anisotropic lengthscales, and the half-normal

distribution is simple, empirically robust, and effective for our applications.

5.4.3 Priors for Lengthscales

Constructing a generic lengthscale hyperprior is slightly more involved than for the marginal standard deviation.

1. **The lengthscale for any given dimension is not much larger than approximately the support of the target function.** Beyond this, functions encoded by the GP are asymptotically linear in that dimension.

2. **The *data* carries no information about lengthscales smaller than the minimum spacing between two points in any given dimension.** Lengthscales smaller than the minimum distance between two points are indistinguishable from one another in the context of their likelihood.

The first consideration can be encoded by selecting a distribution with a sharp right tail, and minimising the probability mass beyond the support of the original function. Since all inputs to our robust model are scaled onto $[0, 1]$, this translates to minimising the probability mass greater than 1.

The second consideration depends on the data. While it goes somewhat against the principles of a "true" Bayesian analysis, an effective method of encoding this consideration is to use the data to construct the prior – a practice sometimes referred to as *empirical Bayes*: by computing the minimum separation of the data in each dimension, we then aim to choose a distribution assigning minimal probability mass less than this value.

Let Δ_i be the minimum separation between any two points in dimension i, and suppose the prior probability of the anisotropic lengthscale in the subscripted dimension, $p(\rho_i|\theta_i)$, is given by a parametric prior distribution, $\pi_{\rho,i}$, with parameters $\theta_{\rho,i}$.

We omit the subscript i herein for clarity and consider it implicit that the analysis

is carried out for each input dimension. We aim to select $\theta_{\rho,i}$ such that:

$$\begin{aligned} \int_0^{\Delta} p(\rho|\theta_\rho)d\rho \leq u \\ 1 - \int_1^{\infty} p(\rho|\theta_\rho)d\rho \leq u \end{aligned} \tag{5.2}$$

given $p(\rho|\theta_\rho) = \pi_\rho(\rho|\theta_\rho)$, and u is a small number representing the proportion of the probability mass which is permitted to be below and above the minimum and maximum spacing of the data.

The system of equations (5.2) can be rearranged to construct a root finding problem in terms of θ_ρ. Letting $\Pi_\rho(\cdot|\theta_\rho)$ be the cumulative distribution function for $\pi_\rho(\cdot|\theta_\rho)$, we hence seek to solve the following for θ_ρ:

$$\begin{aligned} \Pi_\theta(\Delta|\theta_\rho) - u = 0 \\ 1 - \Pi_\theta(1|\theta_\rho) - u = 0 \end{aligned} \tag{5.3}$$

An appropriate parametric form for π_ρ permits sharp left and right tails such that the system of equations described by (5.3) can be adequately satisfied. An effective choice for this purpose is the inverse gamma distribution. We assume the following parametrisation:

$$p(y|a,b) = \frac{a^b}{\Gamma(a)}(\frac{1}{y})^{a+1}\exp[\frac{-b}{y}] \tag{5.4}$$

which has parameters $\theta_\rho \triangleq [a,b]$.

Including the inverse gamma distribution CDF (known in some mathematical programming libraries as the "regularized gamma function") explicitly in the system of equations described by (5.3) results in:

$$\begin{aligned} \frac{\Gamma(a,\frac{b}{\Delta})}{\Gamma(a)} - u = 0 \\ 1 - \frac{\Gamma(a,\frac{b}{1})}{\Gamma(a)} - u = 0 \end{aligned} \tag{5.5}$$

where $\Gamma(\cdot,\cdot)$ represents the incomplete gamma function and $\Gamma(\cdot)$ the regular gamma function.

What remains is to select an appropriate values for u. Smaller values result in a more strict adherence to the two conditions used to construct the prior, as a relatively higher proportion of the prior mass is forced to be assigned between Δ and 1. A small amount of flexibility is usually beneficial in the interest of sampler stability, and we find $u = 0.02$ to be performant in our experiments.

Equation (5.5) is solved for $\theta \triangleq [a, b]$ using a modified Powell method, initialised from a preliminary grid search to locate a suitable starting location. We use `MINPACK`'s `hybrid` solver [45], as implemented in the Python module `SciPy`.

Our lengthscale hyperprior is thus:

$$\rho_i \sim \text{InvGamma}\left(a_i, b_i\right) \quad i = 1, \ldots, d \tag{5.6}$$

where a_i and b_i are identified via the system of equations (5.5) for each input dimension, given Δ_i is the minimum (non-zero) spacing between any two sample points in that subscripted dimension.

5.5 Hyperparameter Inference

In order to satisfy the criteria of a robust model we require an approximation to the full posterior distribution of the model hyperparameters, over which the predictive distribution is to be marginalised.

Let π_θ represent the hyperprior for $\theta \triangleq [\alpha, \rho]$ discussed in the previous two sections. We use MCMC to sample θ from a distribution proportional to $p(\theta|\mathcal{D}, \pi_\theta)$ using Hamiltonian Monte-Carlo (HMC) [30] as implemented by the probabilistic programming language *Stan* [46]. A short introduction to HMC can be found in appendix E.

Stan is a probabilistic programming language facilitating turnkey MCMC sampling

using an enhanced version of the No-U-Turn sampler ("NUTS"), and provides several important advantages relative to other MCMC algorithms which make it suitable for industrial emulation purposes. Most important of these is that the sampling algorithm is parameterless, thus obviating the need for any tuning and safely abstracting away any requirements to rigorously understand sampling behaviour. Slice sampling [28] and methods derived from it are reasonable alternatives for the same reason.

The implementation of NUTS in Stan is particularly convenient, as one need only specify a probabilistic model to run the algorithm, since Stan's automatic differentiation library automatically provides the derivatives of the likelihood function required by NUTS. The process of constructing a probabilistic model in Stan language can be scripted for most relatively simple GP regression problems, which – combined with the metaprogramming techniques mentioned briefly in section 5.3 for constructing covariance kernels based on requested assumptions – means fully defined probabilistic GP regression models can be built automatically for practitioners (critically, in the Stan language) from only a few problem specific considerations.

Since Stan models are translated directly to efficiently programmed C++, compiled Stan models also provide fast sampling via NUTS: on a mid-range laptop with a 2.3GHz Intel i5-6200U, we are capable of drawing 2000 samples of (anistotropic) GP regression hyperparameters in a few seconds for smaller numbers of samples ($N \lessapprox 100$), and in slightly less than a minute for moderate numbers of samples. These figures are even faster for low ($D \lessapprox 5$) dimensional functions, such that NUTS is competitive with likelihood maximisation by multi-restart gradient ascent in terms of clock-time in these cases.

Identical models with larger numbers of samples ($N \gtrapprox 200$) in low dimensions ($D \lessapprox 15$) are noted to be better served by different methods: in these cases there is *often* sufficient information contained in the data such that point estimates of the hyperparameters are acceptable even for complex likelihood functions, and the clock-time difference between NUTS and alternatives such as likelihood optimisation or cross-validation starts to become significant.

5.5.1 Sampler Settings

In general, we use the (default) settings recommended by the Stan developers [21, section 14]: we use 4 independently initialised Markov Chains and draw a total of 1000 samples from each chain, discarding the first half of each chain as "warmup": Since NUTS is an adaptive algorithm, the warmup phase is necessary both to ensure the retained samples are drawn with sampling behaviour that is sufficiently adapted to the geometry of the sampling distribution, and to reduce the influence of the location from which the sample chains are initialised.

Unlike some other MCMC algorithms, the developers report that thinning of the chains (that is, retaining only one every x samples, where x is the amount of thinning) is not usually necessary due to the sampling efficiency of the algorithm, and we follow this recommendation [see also 15, ch. 11 pp. 282-283]. This results in a nominal sample size of 2000 (500 samples drawn from 4 independent chains, after warmup).

By default, NUTS run without "configuration" in the sense that it provides stable and effective sampling performance without requiring any decisions from the user or other context specific information. It does, however, permit optional manual tuning via several parameters, which can help in certain circumstances. Of these, the most relevant in the applications discussed here are the `max tree depth`, and `adapt delta` (which refers specifically to the *target Metropolis acceptance rate*) settings. For the relatively simple GP regression problems we consider, there are good diagnostics for determining when either `adapt delta` or `max tree depth` ought to be changed. These are discussed in section 5.6.

5.5.2 Initialisation

By default, Stan initialises all model parameters randomly on their support [21, section 12.2]. Very occasionally, this can cause issues: if the sampler for a certain chain is initialised in a particularly extreme region of the parameter space, the sampler may be unable to "escape" under its default configuration due to the likelihood in these regions being very flat.

This behaviour is usually indicative of model misspecification (i.e. a model that is incompatible with the supplied data [21, section 21.5]), however can be observed even for simple GP regression problems on occasion if the amplitude is initialised to be small and one or more lengthscales are initialised to be very large. We consider this behaviour in the specific context of GP regression to be innocuous, as it is easily remedied by manual initialisation of the model parameters, and the first half of the sample chains (where the effects of initialisation location ought to be most apparent) are always discarded as warmup, anyway.

Hyperparameters are initialised as follows:

- **the marginal standard deviation**, α, is initialised to the sample standard deviation of the function observations, $\mathbf{y}$. This corresponds to the assuming that the regression function explains all of the observed variation in the data (assuming it has zero mean).
- **anisotropic lengthscales**, $\rho_1, \ldots, \rho_d$, are initialised to one third of the sample standard deviation of the input data in the specified input dimension. This is somewhat arbitrary, though experiments indicate this is a stable initialisation value. In general, any values less than the support of the function of interest and larger than the minimum spacing are usually acceptable.
- **other** parameters are typically initialised to either 0 or 1, depending on their support. Regression coefficients, for instance, are initialised to 0.

We stress that manual initialisation in this manner produces posterior distributions identical to those obtained using random initialisation, except in the rare minority of randomly initialised cases where the sampler becomes stuck.

5.6 Postprocessing

When using MCMC to approximate a distribution, it is important to check the quality of the resulting approximation to ensure that the sampler has run correctly. Such postprocessing can also help to diagnose any incompatibilities between the

model and the data: if the sampler fails to run as it should, this is a possible indication that the model (implicitly, the assumptions used to construct it) and the data are incompatible, and is thus a crude way of assessing the suitability of a particular modelling assumption.

In the following, we follow the recommendations of the Stan developers [21]. Readers are also directed to [15, ch. 11] for a more thorough exposition of sample chain assessment in the context of applied modelling.

5.6.1 Potential Scale Reduction Factor

To assess the convergence of the sample chains, Stan reports an "improved" potential scale reduction factor, $\widehat{R}$ [47], for each of the model's parameters. Loosely speaking, as $\widehat{R} \to 1$ for a particular parameter, this implies additional samples should not contribute strongly to the shape of that parameter's marginal distribution, thus indicating that the distribution has "converged" to a stable shape.

Observing $\widehat{R} \geq 1.01$ is a reason to believe further simulations are necessary to resolve the shape of the posterior (marginal) distribution for that particular parameter, so the sampler should be run again with a higher number of iterations.

5.6.2 Divergences

The Stan implementation of NUTS reports several sampler diagnostics. Of particular interest is the possible presence of divergent transitions, which may indicate the shape of the posterior distribution cannot be trusted.

In basic terms, divergent transitions are caused by highly curved posterior geometry (i.e. areas where the likelihood function has steep gradients) that the sampler fails to navigate reliably [48]. In some cases, this can be resolved by increasing the target acceptance rate (the `adapt delta` parameter mentioned in section 5.5.1), while in others a re-parametrisation of the model is necessary (See experiments with *Neal's Funnel* [49] and [21, section 20.2]).

For the relatively simple GP regression models considered here, experience suggests

that divergent transitions are almost always be resolved by increasing `adapt delta`. Our models do not appear to induce any particularly pathological posterior geometry, and all encountered divergences during the later stages of development were resolved by gradually increasing `adapt delta`.

Presence of any divergent transitions in the experiments considered in this work is reason to slightly increase `adapt delta` (from it's default value of $0.8 \rightarrow 1$, in small increments) and restart the sampler. This is easily automated. If the value of `adapt delta` becomes very close to 1 and divergences remain, this suggests an issue with the parametrisation of the model or a problem with the data.

5.6.3 Tree Depth Saturation

NUTS also reports tree-depth saturations if the number of leapfrog steps (see appendix E) reaches a pre-specified limit (the *max tree-depth*), which has a slightly increased tendency to occur if `adapt delta` is increased.

Unlike divergences, tree depth saturations indicate inefficient exploration of the posterior rather than potentially incorrect exploration, though encountering tree-depth saturation warnings is considered reason to increase the `max tree depth` parameter and restart the sampler due to the potentially diminished effective sample size.

5.7 Summary

We introduce the concept of a "robust emulator": a statistical model providing consistently reliable quantification of predictive uncertainty. To enable easy construction of these models, we discuss generally applicable prior distributions and the use of parameterless MCMC sampling via the probabilistic programming language *Stan* to provide a fully probabilistic treatment of the model hyperparameters and a more complete representation of uncertainty implicit in the model fit procedure relative to other methods.

Chapter 6

Experiments for Robust Emulators

We test the performance of the robust emulators introduced in the previous chapter on a variety of synthetic and analytical engineering test functions against a range of similar models, showing favourable performance of the robust models in terms of their ability to consistently represent predictive uncertainty accurately, in addition to competitive point predictive performance.

6.1 Hypothesis

Our hypothesis is that by constructing a robust model, which includes a fully probabilistic representation of any model fit uncertainty, we will obtain a more complete and reliable representation of the uncertainty associated with the use of the emulator.

In many industrial surrogate modelling applications, we anticipate that the information supplied by the data alone is not sufficient to result in a hyperparameter distribution that is *always* well summarised by a single point: this is due to the cost of data acquisition associated with expensive physics simulators (such as the loads process discussed earlier in this document), which place an upper limit on the volume of data that can be obtained in practice. How "disperse" this hyperparameter distribution is becomes more severe with increasing data "sparsity", which in the context of a specific model can be considered qualitatively to be a function of the quantity of data, the number of dimensions across which it is spaced, and the complexity of the model.

In this sense, many industrial surrogate modelling applications in aerospace engineering are sparse to moderately sparse. We assert that robust models, which use the full posterior distribution of the model hyperparameters, will exhibit more stable predictive performance that is less prone to specifics of the data.

Quantifying the relationship between model complexity, data volume and hyperparameter "identifiability" is an interesting and challenging avenue for further investigation but is not considered in this document.

6.2 Method

We assess the performance of emulators fit using three different methods:

- A robust method, as discussed in chapter 5, fit using **MCMC**.
- A maximum likelihood estimate using flat hyperpriors ("no prior"). We will refer to this method as **MLE**.

- A maximum likelihood estimate of the hyperparameters using the hyperpriors discussed in section. We will refer to this method as **MLE-II**.

We use the canonical GP regression model consisting of a zero-mean function and an exponentiated quadratic (EQ) kernel which is a suitable first choice for the test functions we consider for the reasons outlined briefly in chapter 2. Our covariance kernel is:

$$k(\mathbf{x}, \mathbf{x}'|\theta) = \alpha^2 \text{exp}\left[-0.5 \sum_{i=1}^{d} \frac{(x_i - x_i')^2}{\rho_i^2}\right] \tag{6.1}$$

with hyperparameters summarised by $\theta \triangleq [\alpha, \rho_1, \ldots, \rho_d]$.

We also add stability jitter with magnitude $\sigma^2 = 1e^{-10}$ to the leading diagonal of the covariance matrix for numerical stability. This value ensures the resulting covariance matrix is always invertible (that is, it is positive definite), but is sufficiently small so as not to have any influence on the models' predictive performance.

For the optimisations, we maximise the log-likelihood (log marginal likelihood in the case of the MLE-II method) using Stan's implementation of Newton's method, which makes use of automatically computed derivatives of the chosen probabilistic model. Convergence is assumed when the change in the objective function between iterations is less than $1e^{-10}$, and the optimisations are repeated 5 times, with the result with the highest log (marginal) likelihood returned.

Detailed documentation of the fit procedures for each of the three models, including probabilistic models, is provided in appendix F.

6.3 Assessing Probabilistic Predictive Performance

Assessing a faithful representation of *uncertainty* is not as straightforward as assessing predictive performance for point predictions. Measuring predictive accuracy for probabilistic predictions is discussed in [15, ch. 7].

6.3.1 Log Pointwise Predictive Density

We opt to use the Expected Log Pointwise Predictive Density (ELPPD). In simplified terms, the ELPPD rewards correct predictions proportionally to how confidently they are made, and penalises incorrect predictions proportionally to how confidently they are made.

Let y^* represent a single response from the true data generating process, $\mathbf{x}^*$, such that $y^* = f(\mathbf{x}^*)$. The Log Predictive Density (LPD) for a single point is:

$$\text{lpd} = \log\left(\int p(y^*|\mathbf{x}^*, \mathcal{D}, \theta)p(\theta|\mathcal{D}, \pi_\theta)d\theta\right) \tag{6.2}$$

where $p(\theta|\mathcal{D}, \pi_\theta)$ is the posterior probability of the hyperparameters and $p(y^*|\mathbf{x}^*, \mathcal{D}, \theta)$ represents the predictive distribution of the model (see appendix B).

The posterior distribution of the hyperparameters, $p(\theta|\mathcal{D}, \pi_\theta)$, is unknown in practice and is approximated either by sampling from an approximation to it (MCMC), or condensation to a single point (MLE / MLE-II). The integral with respect to θ in equation (6.2) can thus be replaced by the expectation over S samples:

$$\text{lpd} = \log\left(\mathbb{E}\left[p(y^*|\mathbf{x}^*, \mathcal{D}, \theta)\right]_\theta\right) = \log\left(\frac{1}{S}\sum_{s=1}^{S} p(y^*|\mathbf{x}^*, \mathcal{D}, \theta_s)\right) \tag{6.3}$$

The Log *Pointwise* Predictive Density (LPPD) for a *dataset* of size N involves summing the log predictive densities described by equation (6.3) for each of the N points:

$$\text{lppd} = \sum_{i=1}^{N} \text{lpd}_i = \sum_{i=1}^{N} \log\left(\frac{1}{S}\sum_{s=1}^{S} p(y_j^*|\mathbf{x}_j^*, \mathcal{D}, \theta_s)\right) \tag{6.4}$$

In the case of the MLE and MLE-II models specifically, equation (6.4) collapses to computing the sum of the log predictive densities of the observed data as only a

single sample of θ is present:

$$\text{lppd} = \sum_{i=1}^{N} \log \left(p(y_j^* | \mathbf{x}_j^*, \mathcal{D}, \theta) \right) \tag{6.5}$$

We scale equation (6.4) and equation (6.5) by $1/N$, which equates to computing the *Expected* Log Pointwise Predictive Density (ELPPD) for an arbitrary point selected at random from the dataset.

A high ELPPD score on a particular dataset shows that the emulator deems those observations as having high probability, or more loosely, "was not surprised" by the data: this is desirable as it is a good indication that the model has done a good job of modelling uncertainty present in the dataset. Conversely, a low ELPPD score indicates the emulator was "surprised" by the data, or more specifically that some of the observations have low or very low probability according to the model.

6.3.2 Root Mean Square Error

A potential concern when using probabilistic measures of performance such as the ELPPD is that the model may behave in an overly conservative fashion, with reasonably high ELPPD scores disguising poor point-predictive performance. To alleviate such concerns, we also report the Root Mean Square Error (RMSE) for each of the models via the expectation of the predictive distributions (i.e. the GP predictive mean, see chapter (2) or appendix B).

6.3.3 Model Self Assessment

In many surrogate modelling applications, practitioners do not have the luxury of a large test set of data on which to evaluate performance. A model which is capable of evaluating its own performance reliably via (for example) cross validation is consequently valuable, as a larger portion of the simulation budget can be allocated for training.

In the case of GP emulators, this results not only in more information being available

to identify the model hyperparameters, but also more observations on which to condition the model, which provide a reduction in predictive uncertainty in their vicinity.

Using the method of Sundararajan and Keerthi [23] introduced in chapter 3, we compute the analytical leave-one-out predictive densities for our models, and take the expectation of this quantity across N_{test} training samples to provide an estimate of the ELPPD score on the test set. We then report the absolute difference between the cross validated ELPPD and the actual ELPPD on the test data. Restating 3.15 from section 3.4.2, the leave-one-out predictive densities for a zero-mean GP are calculated by:

$$\text{LooCV}(\theta|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N_{\text{train}}} -0.5\log(2\pi) + 0.5\log(\omega_{\theta,i}) - 0.5(\frac{\alpha_{\theta,i}^2}{\omega_{\theta,i}})$$

where, $\Omega_\theta = [K_{\theta,\mathbf{XX}}]^{-1}$; $\omega_\theta = \text{diag}(\Omega_\theta)$; and $\alpha_\theta = \Omega_\theta \cdot (\mathbf{y})$

6.4 Analysis Procedure

Each model is fit to a variety of different test functions using datasets of gradually increasing size. The performance of the models is then assessed on a large, randomly generated test set. These experiments are repeated 50 times each to explore the sensitivity of the fitting mechanisms to slightly different sets of training data and the resulting effects on predictive performance.

6.4.1 Functions

It is critical that the test functions can be evaluated rapidly, as they are queried many thousands of times over the testing procedure. We use a variety of synthetic functions and analytical physical models commonly used as emulation test problems sourced from *The Virtual Library of Simulation Experiments* [50]. This provides us with "engineering like" test functions which are representative of industrial modelling problems while being fast to evaluate.

Name	D	**Reference**
Branin	2	[51]
Cosines	2	[52]
Hypersphere	4	See appendix F.4.2
6-hump Camel	2	[53]
Friedman	5	[54]
Piston	7	[55]
Borehole	8	[56]
Circuit	6	[55]
Wingweight	10	[57]

Table 6.1: Summary of test functions, corresponding input dimensions, and references.

All test functions are adapted to take inputs on $[0, 1]^D$ and are summarised by table 6.1. Surface plots of the 2D functions – Branin, cosines, and 6-hump camel function – are provided in appendix F.

6.4.2 Training Data

We analyse each test function at training sample sizes of varying sparsity. Training samples of a given size are generated using the min-max Latin-Hypercube Sample (LHS) design of Morris and Mitchell [40]. Since min-max LHS is widely used in industrial surrogate modelling applications, it is representative of a realistic use case, and is sufficiently "randomised" such that repeated realisations can be used to probe different model's sensitivities to specifics of the training sample.

In accordance with the rationale described in chapter 5, the test data is scaled to approximately unit standard deviation and zero mean.

6.4.3 Test Data

We make use of large ($N_{\text{test}} = 600$) test sets to evaluate the ELPPD and RMSE. This is an arbitrarily large number of test points, used to provide dense coverage. To ensure the points are uniformly distributed – thus ensuring our evaluations do not inadvertently weight performance in certain regions of the input space over others – we generate quasi-random space-filling sequences on $[0, 1]^D$ using the Sobol sequence [58], which has been demonstrated both theoretically and empirically to

exhibit strong multi-dimensional space-filling properties [59].

The test data is normalised using the same factors as the appropriate training sample.

6.5 Results

Values of ELPPD and RMSE are reported for each test function, for varying sample sizes. Each type of model is fit and evaluated 50 times for each test function and sample size. Results for the test functions are displayed in figures (6.1 - 6.4) for the 5 synthetic functions (Branin, Cosines, Sphere, 6-hump Camel and Friedman) and in figures (6.6 - 6.9) for the 4 engineering functions (Piston, Circuit, Borehole, Wingweight).

We also report the log difference between the leave-one-out cross validated ELPPD and the ELPPD computed on the test data for each test function in figure 6.10.

6.5.1 Branin

With reference to figure (6.1), ELPPD can be seen to increase, on average, with sample size, and rmse can be seen to decrease. MCMC and MLE-II perform favourably in terms of ELPPD, with MLE experiencing a large amount of variation, particularly at lower sample sizes. MLE-II can be observed to exhibit slightly inconsistent ELPPD performance at higher N, though these points are in general outliers.

6.5.2 Cosines

With reference to figure (6.2), MLE is noted to suffer from strong overconfidence to the extent that its ELPPD scores are infinitely low (thus not visible). MCMC appears to exhibit superior ELPPD performance relative to MLE-II but exhibits similar RMSE. Poor ELPPD performance of MLE, on average, is likely due to the linear bias induced by a flat prior and the fact that the target function is highly nonlinear.

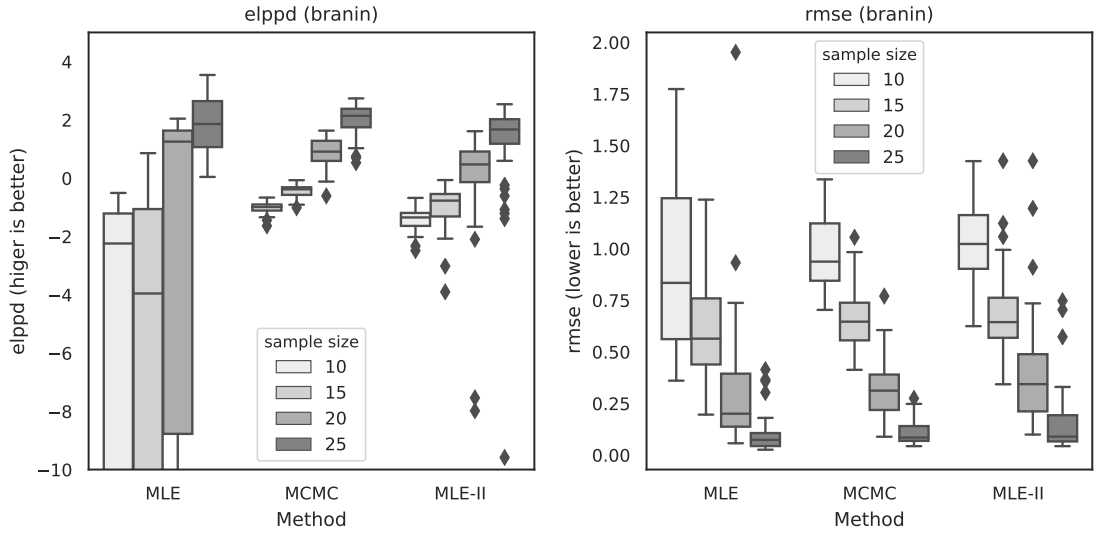

Figure 6.1: ELPPD and RMSE for the Branin function.

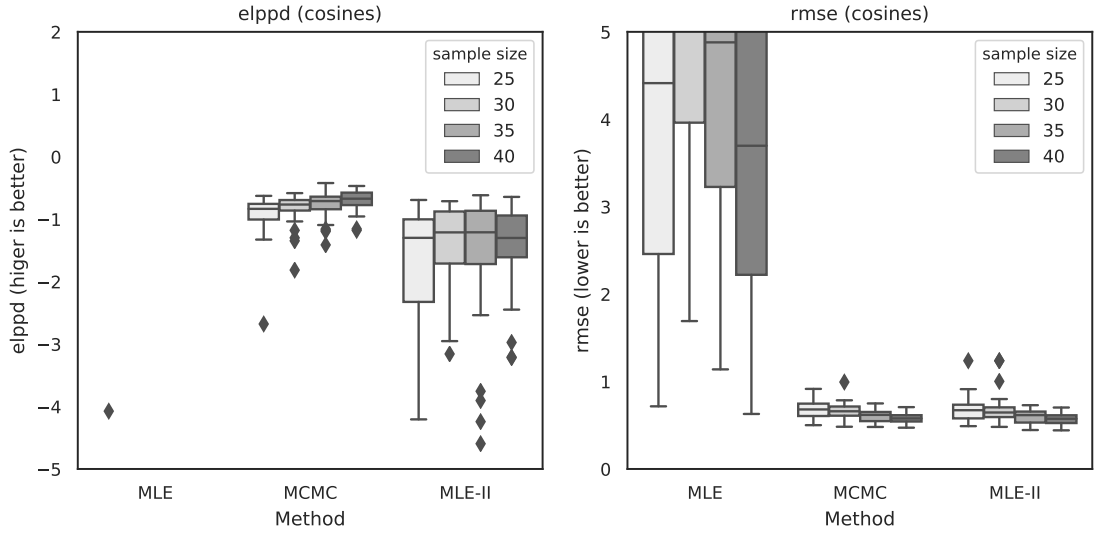

Figure 6.2: ELPPD and RMSE for the Cosines function.

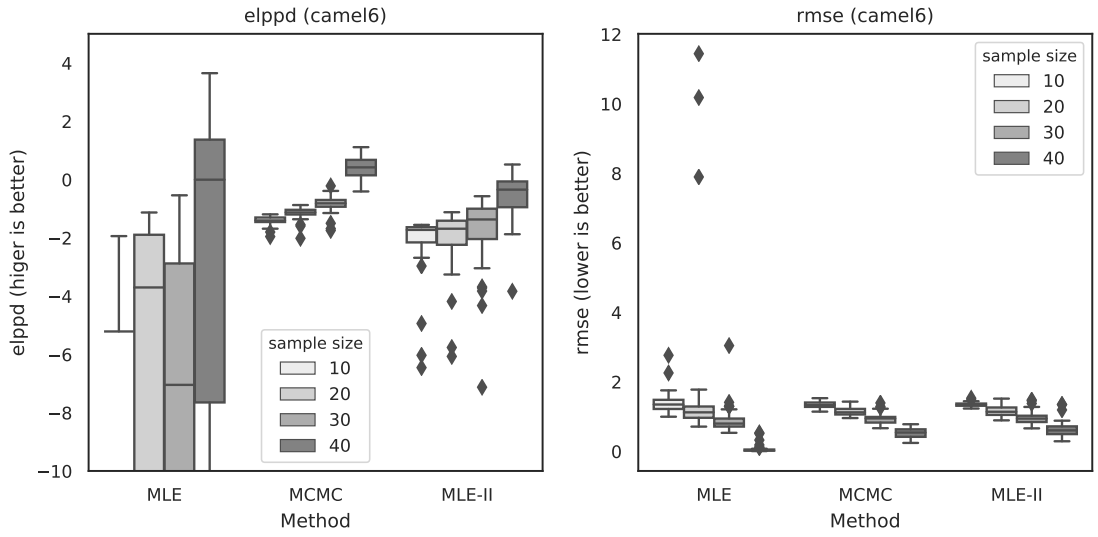

Figure 6.3: ELPPD and RMSE for the 6-hump camel function.

6.5.3 6-Hump Camel Function

Referencing figure (6.3), MLE exhibits poor ELPPD performance, though only slightly worse RMSE performance than MCMC at higher ($N > 15$) sample sizes, and best average RMSE at low sample sizes. MCMC performs well in terms of ELPPD, though somewhat more inconsistently than on other functions, and favourably in terms of RMSE, except at low N. MLE-II performs only slightly worse than MCMC in terms of ELPPD, on average, though with a large amount of variation between samples, and similarly to MCMC in terms of RMSE.

6.5.4 Friedman Function

Figure (6.4) shows the results for the Friedman test function. Though MLE is unstable at low N, it performs well on this function in terms of RMSE, and well in terms of ELPPD, except between $N = 40$ and $N = 50$ samples, at which point a spurious reduction in ELPPD can be observed. MCMC performs similarly, on average, but with much more consistency, and does not experience a drop in ELPPD performance when moving from $N = 40$ to $N = 50$. MLE-II performs more favourably and with more consistency than MLE in terms of ELPPD, but not RMSE, except at $N = 50$, and in general has slightly inferior performance relative to MCMC in terms of both ELPPD and RMSE.

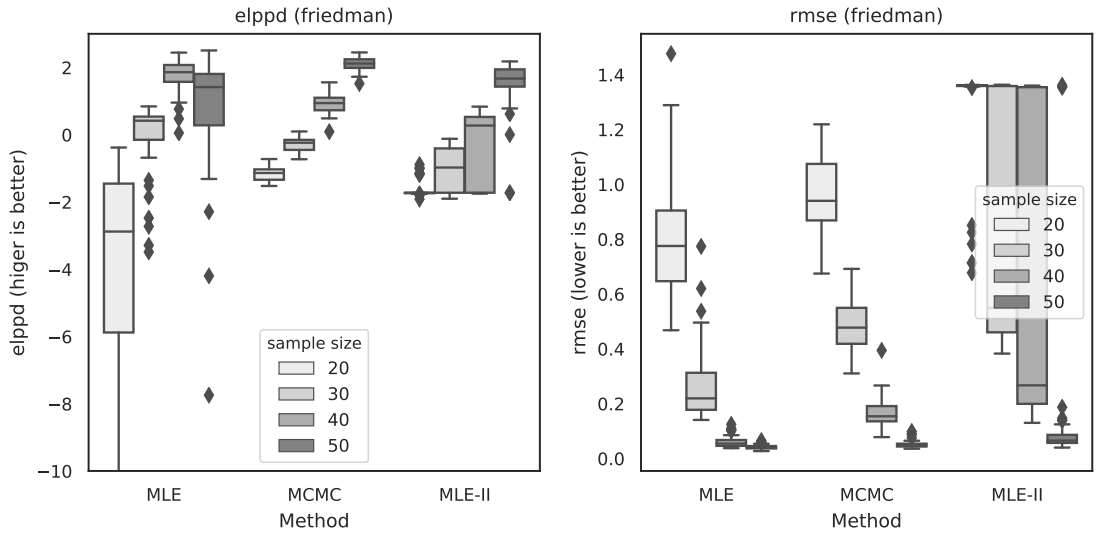

Figure 6.4: ELPPD and RMSE for the Friedman function.

6.5.5 4D Sphere

With reference to figure (6.5), MLE performs inconsistently on the 4D Sphere but occasionally with the best performance in terms of both ELPPD and RMSE. MLE appears to be consistently overconfident at higher sample sizes, characterised by good RMSE performance but poor ELPPD performance. Notably, the distribution of ELPPD results for MLE appears to be multi-modal at $N = 30$. MCMC and MLE-II exhibit similar performance, with MCMC demonstrating slightly superior performance in terms of ELPPD and RMSE at higher sample sizes.

6.5.6 Piston

The results for the piston function are shown on figure (6.6). MLE exhibits inconsistent ELPPD performance that increases in quality with sample size. The ELPPD performance of MCMC is consistently superior and exhibits less sensitivity to different datasets. Both MLE and MCMC perform very well in terms of RMSE, with MLE being superior at low sample sizes. MLE-II can be seen to perform poorly in all cases: despite having higher ELPPD than MLE at low sample sizes, this is offset by poor RMSE performance.

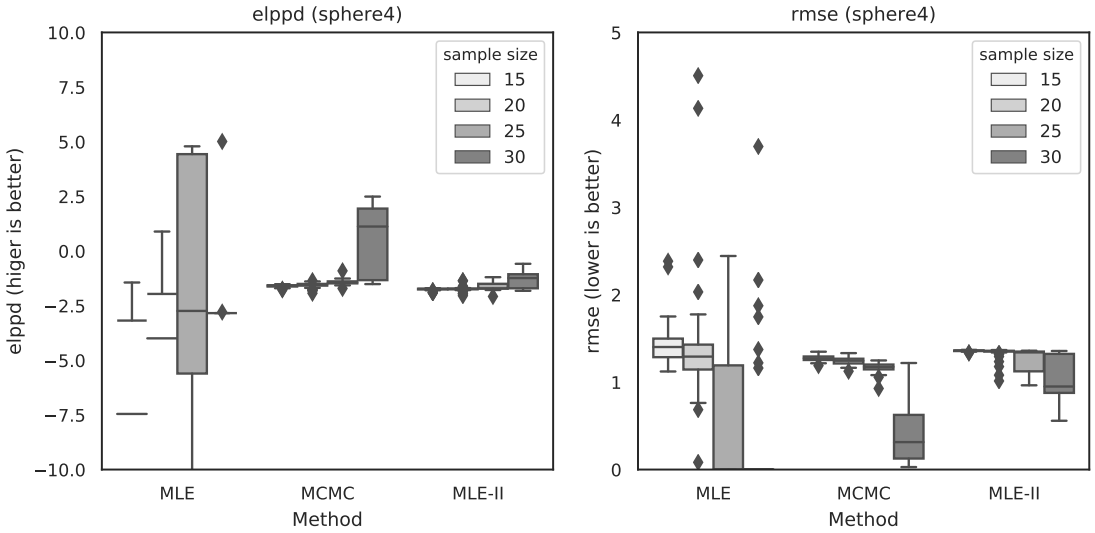

Figure 6.5: ELPPD and RMSE for the 4D hypersphere function.

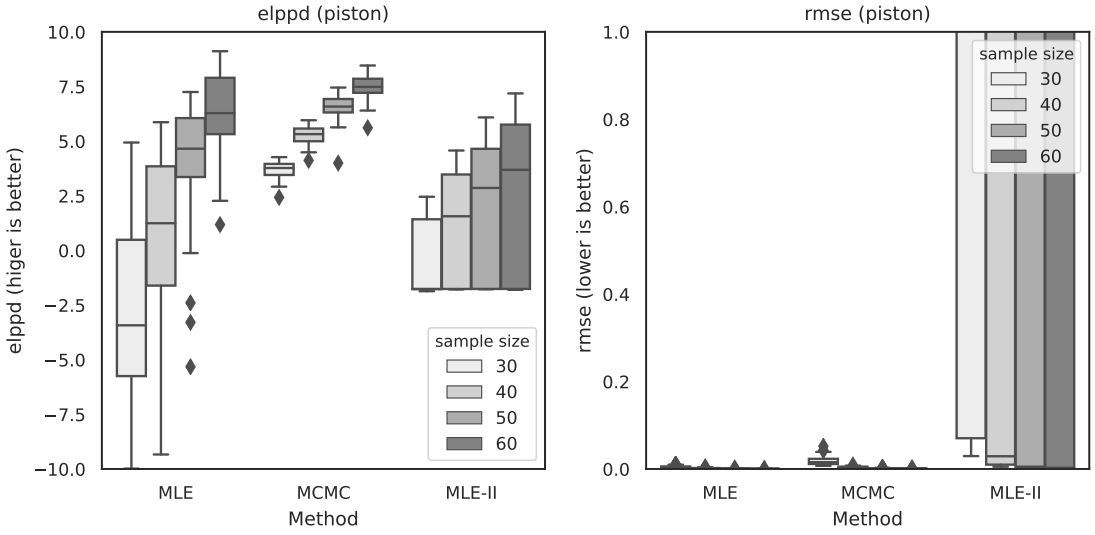

Figure 6.6: ELPPD and RMSE for the piston function.

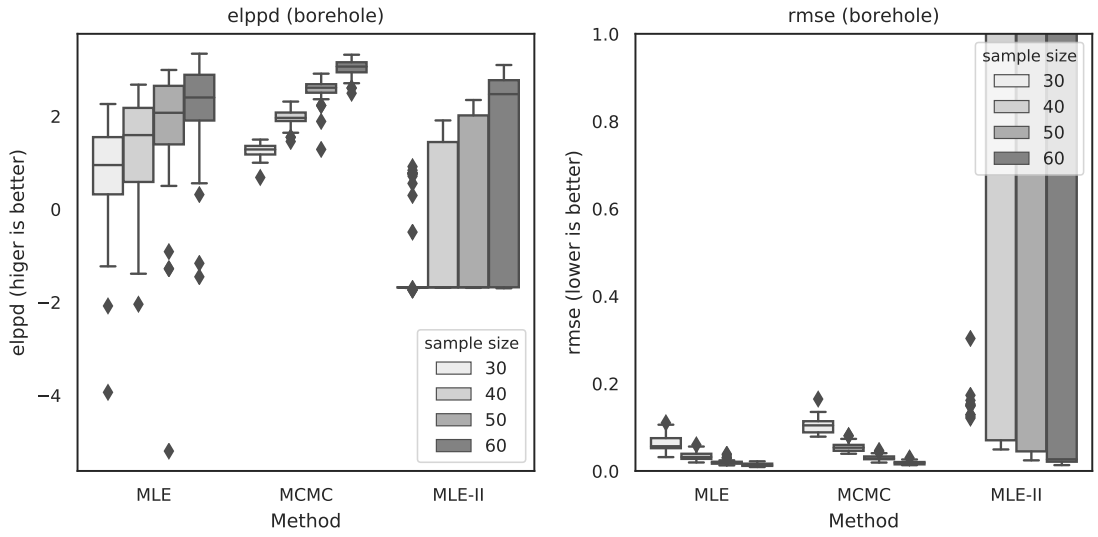

Figure 6.7: ELPPD and RMSE for the borehole function.

6.5.7 Borehole Function

Figure (6.7) shows the results for the borehole test function. MLE-II can be seen to exhibit inconsistent performance in terms of both ELPPD and RMSE, which is on average worse than MLE and MCMC. MCMC offers consistently higher ELPPD performance than MLE, though MLE offers slightly better RMSE performance, on average. These differences appear to reduce with sample size.

6.5.8 Circuit Function

Figure (6.8) shows the results for the circuit test function. MLE exhibits poor and inconsistent ELPPD performance, though good RMSE performance. MLE-II fares slightly better than MLE in terms of ELPPD, but has inconsistent ELPPD performance. MCMC consistently performs the best in terms of ELPPD and has performance slightly worse than MLE in terms of RMSE.

6.5.9 Wing Weight Function

Figure (6.9) shows the results for the wing weight test function. MLE exhibits high sensitivity to different realisations of the data in terms of ELPPD, though performs well in terms of RMSE, with one notable outlier at a sample size of 30. The MLE

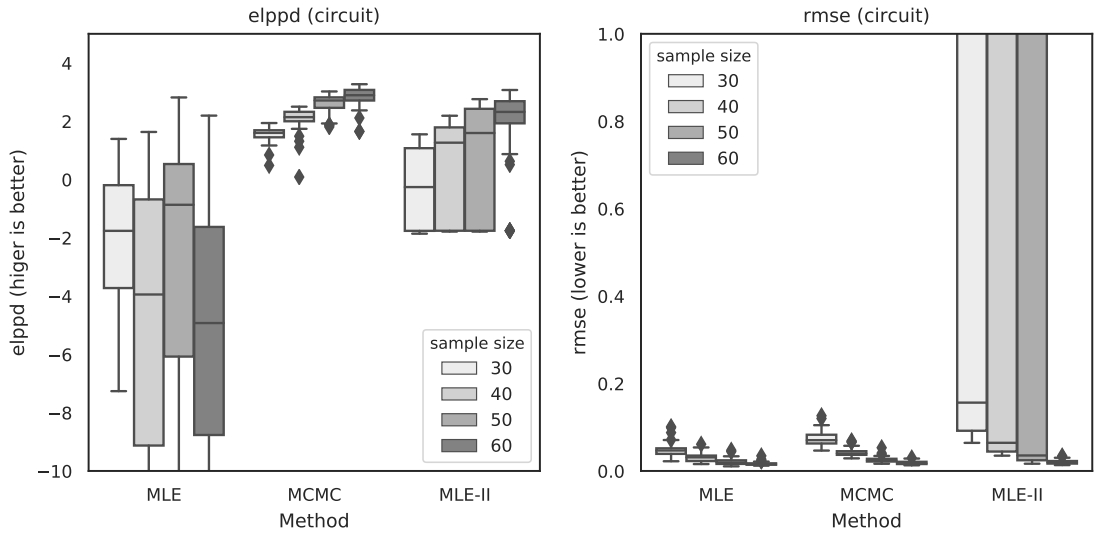

Figure 6.8: ELPPD and RMSE for the circuit function.

models also exhibit unusual ELPPD behaviour, in which average ELPPD decreases with sample size between $N = 30$ and $N = 50$. MCMC performs consistently well in terms of ELPPD and RMSE, exhibiting the best performance in both except in terms of RMSE at high sample sizes, in which case the median performance of MLE is superior. MLE-II appears to encounter difficulties with this function, exhibiting multi-modal performance characteristics in terms of ELPPD, and highly inconsistent RMSE (with box plots not shown at $N < 60$ due to being outside the figure axis).

6.5.10 Model Self Assessment Scores

Figure (6.10) shows the log absolute differences between the Leave-one-out cross validated ELPPD scores and the ELPPD scores obtained using test data. The *log* difference is reported so that very small and very large differences can be reported using the same axis. MLE-II quite consistently exhibits the lowest differences between the two criteria, though MCMC is comparable on some test functions (branin, circuit, wingweight), and the differences tend to be small. MLE consistently demonstrates the largest differences.

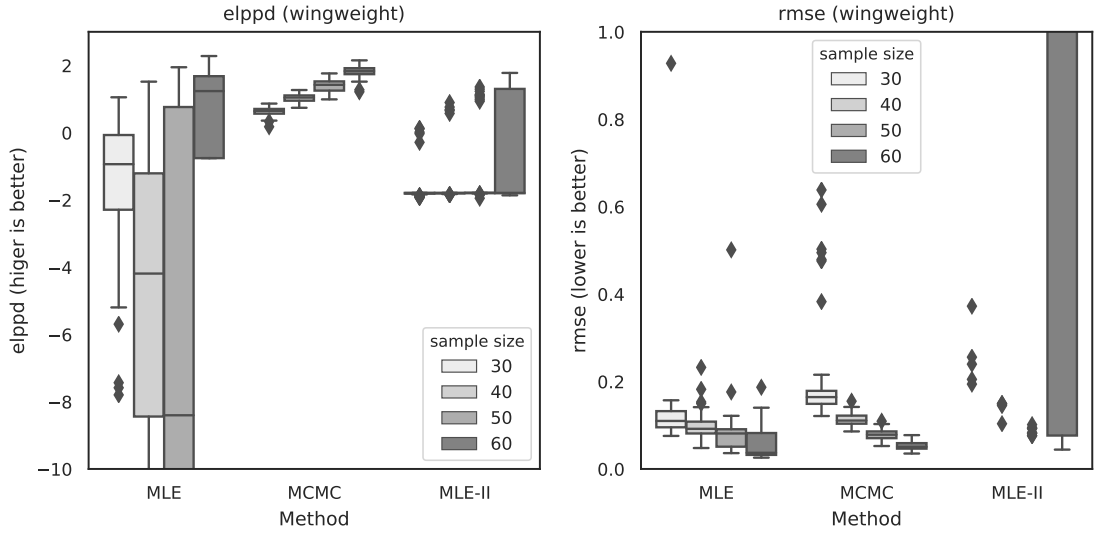

Figure 6.9: ELPPD and RMSE for the wing weight function.

6.6 Analysis

On balance, the robust models fit using MCMC perform favourably, exhibiting good though not necessarily *best*) ELPPD and RMSE performance on both the synthetic (Branin, Cosines, Hypersphere, 6-hump Camel, Friedman) and engineering (Piston, Circuit, Borehole, Wingweight) test functions. The MCMC method also appears to be significantly less sensitive in terms of ELPPD to different initial random samples, which we consider to be due to the extra stability introduced by our choice of hyperprior distributions and the fact that the models are integrated over uncertainty in the hyperparameters. These observations confirm our initial hypothesis that the MCMC method should perform more consistently and provide a more reliable representation of uncertainty associated with use of the model.

The MLE method exhibits inconsistent ELPPD, indicating heightened sensitivity to the initial sample points relative to MCMC in terms of the method's ability to represent uncertainty in its predictions. The method exhibits good, and often best, RMSE performance, particularly on the engineering functions, figures (6.6-6.9); and for the synthetic functions (6.1-6.4) at larger sample sizes, though poor ELPPD, especially on the engineering functions. Low RMSE *and* low ELPPD is characteristic of model overconfidence, occurring when the target is *almost* but not quite linear

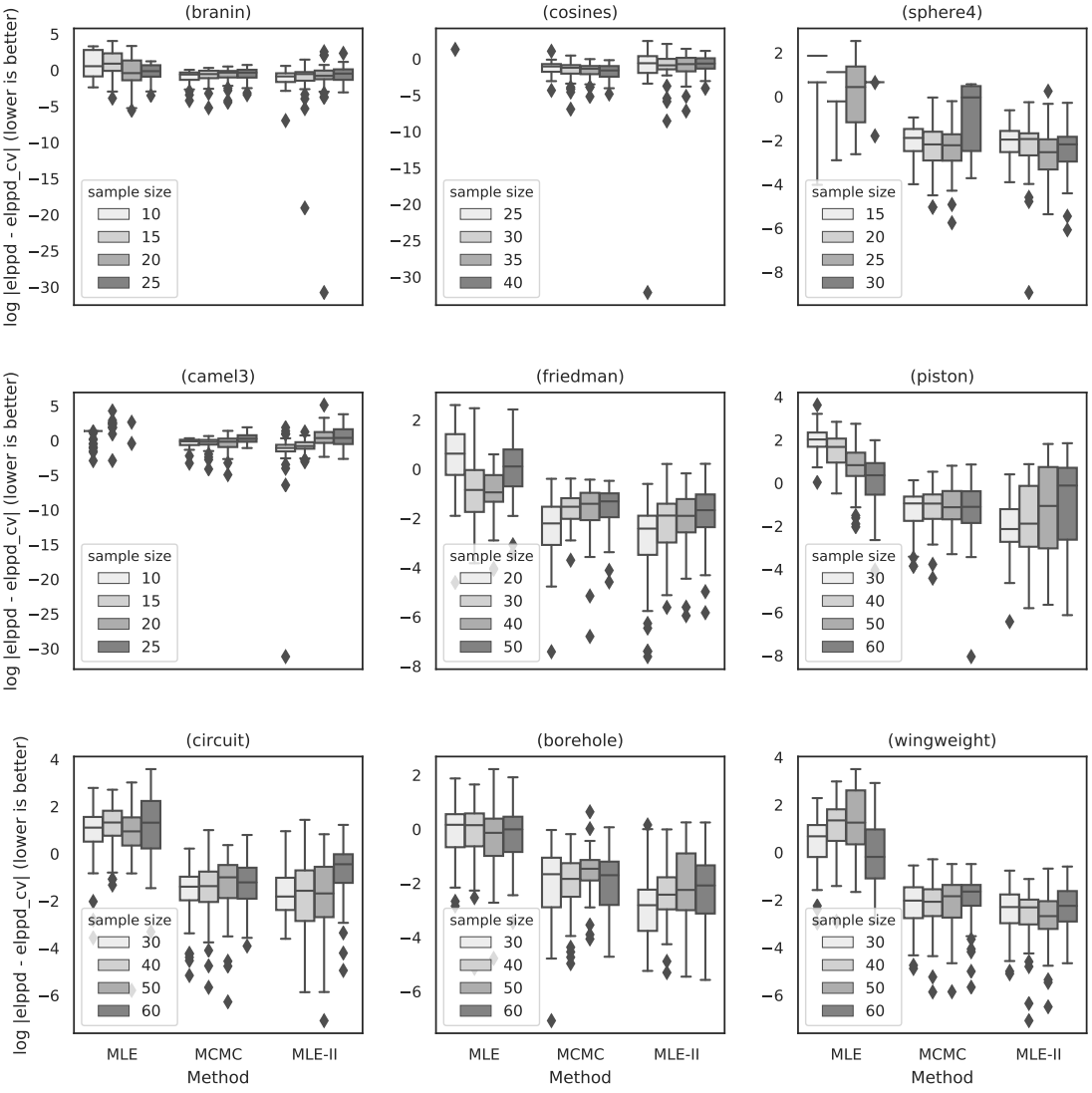

Figure 6.10: Log absolute differences between the leave-one-out cross validated ELPPD scores and the ELPPD scores obtained on the test data.

across its support, perhaps lacking information from the data such that this slight nonlinearity can be considered by the likelihood function. This appears to be the case concerning the engineering functions, and inspecting the kernel lengthscales for the models reveals many are linear ($\rho > 2$) in most of their input dimensions.

This same behaviour partially explains the degraded RMSE of the MLE-II model on the engineering functions relative to MLE, as the chosen hyperprior shifts the maximum likelihood estimate (as regularized by the prior, in this case) towards more nonlinear lengthscales; *and* why MLE-II performs more favourably on the synthetic functions in figures (6.1-6.4), since these tend to be more nonlinear with respect to their inputs. The low RMSE observed by the MLE-II models on the engineering functions in figures (6.6-6.9) is likely due to the more nonlinear models being a poorer prediction for the data, as indicted by the observed ELPPD scores being low but occasionally better than those for the MLE models, which otherwise have superior RMSE scores. This behaviour is not exhibited by the MCMC models, which despite receiving identical prior distributions to the MLE-II models, are integrated over the full hyperparameter distribution, shifting the resulting integrated model away from an overly nonlinear representation for the data.

There are occasions where the point estimate models (MLE and MLE-II) exhibit "multi-modal" performance characteristics, visualised as dense clusters of outlying points on the box plots. This can be observed particularly for MLE model ELPPD on the sphere function, figure (6.5), and MLE-II ELPPD on the wing weight function, figure (6.9). The fact that this behaviour is never seen for MCMC suggests this could be due to a multi-modal or ridge shaped distribution of one or more important hyperparameters, such that point estimates are not effective summaries in these cases.

All of the methods generally improve in quality with more data – as expected – though this is occasionally more dramatic in certain circumstances: in particular, the average ELPPD scores of the MLE method on the Branin function between 20 and 25 samples, observed in figure (6.1); and on the wing weight function between 50 and 60 samples, observed in figure (6.9), appear to experience significant improve-

ment. This may suggest some kind of sample-size threshold over which the data is capable of providing enough information such that the model hyperparameters are well identified, in particular providing enough information that the overly linear lengthscales of the MLE models (represented by good RMSE but poor ELPPD) can be shifted to more appropriate nonlinear values.

A notable exception occurs in the case of MLE model ELPPD scores, which on several occasions become *worse* with more data: notable instances of such behaviour are the Friedman function between $N = 40$ and $N = 50$, shown in figure (6.4); the circuit function at all sample sizes, shown in figure (6.8); and the wingweight function between $N = 30$ and $N = 50$, shown in figure (6.9). In each of these cases, reductions in average ELPPD are coupled with increased average RMSE, indicating this phenomenon is to do with poorly quantified predictive uncertainty rather than predictive performance more generally. This is indicative of unsuitable model hyperparameters. As this behaviour is not apparent for either the MCMC or MLE-II models, one possible explanation is that the linear bias originating from use of a flat lengthscale prior results in insufficiently generous predictive uncertainty, with this overconfidence apparently exacerbated by certain datasets: if one (or more) samples do not lie in particular (problematic) regions of the input space, an increased volume of data which otherwise may be consistent with an overly linear model will increase confidence in such overly linear hyperparameters. This explanation is consistent with the fact that though *average* ELPPD decreases, *maximum* ELPPD tended to either improve or remain the same.

MCMC and MLE-II appear to be the best at evaluating their predictive performance via LOO-CV, as seen on figure (6.10), having the lowest absolute differences between the ELPPD scores on the test data and the analytically computed ELPPD estimates via leave-one-out cross validation.

6.7 Summary

Robust emulators, as introduced in the previous chapter, have been rigorously tested against models fit using both Maximum Likelihood Estimation and prior-regularized Maximum Likelihood Estimation (MLE-II), with the robust models exhibiting favourable performance. In particular, the robust models exhibited significantly less sensitivity in terms of predictive performance to specifics of the data used to train them.

Chapter 7

Critical Load Case Selection

Revisiting the loads process, we investigate uncertainty associated with the choice of critical load cases. In this chapter, we approach a slightly simplified problem applicable to situations in which anticipated changes to the design are small. We show how Bayesian Optimisation can be used as a simulation-efficient means of identifying the critical load cases, and how to construct a probability distribution over possibly-critical cases such that a quantitative load case downselection is feasible.

7.1 Problem Description

In chapter 4, we introduced the loads process as a black-box function, $l(\mathbf{x}, \mathbf{z})$; used to calculate aircraft loads as a function of a load case, $\mathbf{x}$; and a design vector, $\mathbf{z}$, returning a vector, $\mathbf{y}$, of forces and moments at locations of interest (or "nodes") on the airframe:

$$\mathbf{y} = l(\mathbf{x}, \mathbf{z}) \tag{7.1}$$

A particular load case, $\mathbf{x}$, was determined to be *critical* (denoted by superscript *), if:

$$l(\mathbf{x}^*, \mathbf{z}) \geq l(\mathbf{x}, \mathbf{z}) \quad \forall \mathbf{x} \in \mathcal{X} \tag{7.2}$$

holds for at least a single IQ $y \in \mathbf{y}$.

This can be posed neatly as a maximisation problem over the load case space:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} l(\mathbf{x}, \mathbf{z}) \tag{7.3}$$

where $\mathbf{x}^*$ is a critical load case.

The optimisation described by equation (7.3) is noted to be challenging, due to the size of the load case space, $\mathcal{X}$, and the cost associated with querying the objective function. This is to the extent that $\mathcal{X}$ tends to be discretised using domain expert knowledge and/or past data, and the critical cases estimated from a discrete set. This, however, produces uncertainty in the computed enveloping loads, since the location of the true critical cases for any given design is never directly investigated.

This chapter aims to provide tools to investigate this uncertainty, first proposing a strategy for approaching a simplified version of the optimisation problem in a sample efficient manner given relatively mild but assumptions, and then proposing

a method for representing uncertainty in the location of the critical cases.

7.2 Critical Case Identification for a Fixed Design

We first approach a slightly simplified version of the full problem by assuming $\mathbf{z}$ is fixed. This is equivalent to assuming the design vector either does not change, or experiences only small changes in the context of the resulting loads. This is neither an unrealistic assumption nor a toy problem: the same rationale is used to justify discretisation of the load case space when performing case downselection as discussed in [2], and the critical cases of designs close to a baseline configuration ought not to change much. Additionally, such an analysis can be used to investigate the uncertainty associated with a particular choice of critical cases, albeit only for a single design.

Critical case identification for a fixed design is described by the following optimisation over the load case space:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} l_{\mathbf{z}}(\mathbf{x}) \tag{7.4}$$

where the subscript $\mathbf{z}$ is used to note that the objective function is conditional on the choice of design.

The optimisation described by equation (7.4) is still challenging: The load case space is difficult to navigate, the objective function is expensive, and the optimisation must be repeated (in principle) for each IQ.

Our strategy for approaching this problem involves the use of a data-sparse emulator to provide a model $l_{\mathbf{z}}(\mathbf{x})$, followed by subsequent adaptive sampling of the load case space *only* in regions that are informative about the critical cases. Such a strategy is functionally equivalent to performing *Bayesian optimisation* (BO).

7.3 Bayesian Optimisation For Loads

Use of a GP as an emulator offers the possibility of deploying an adaptive data acquisition strategy to the construction of the model. This is useful from an industrial perspective as it ensures expensive simulation resources are allocated only where they are useful (in some formal sense) and necessary. The intuition behind BO is introduced in appendix G and an accessible technical review can be found in [60]. Since BO is a function of the predictive *distribution* of the emulator, a model which confers a good representation of predictive uncertainty is advantageous. This assertion is confirmed by experiments conducted by Hernández-Lobato et al. [61]. We consequently make effective use of the robust GP regression techniques introduced in chapter 5.

In the context of critical load case identification, we wish to use BO as a strategy for (sequentially) investigating only those load cases which contribute knowledge about the location of the critical cases. Practically, this involves first constructing an emulator using a low number of initial load cases, then sequentially investigating more load cases deemed to be "useful" via an *acquisition function*.

7.3.1 Mock Loads Process

Testing this procedure on the real loads process is impractical due to the amount of time required to perform a query: in order to marginalise how much the initial, random sample influences the results of our BO procedure, the experiments must be repeated a considerable number of times. Given the loads process is expensive to work with (in the sense that each query requires a large amount of time to be evaluated), repeating this procedure even several times is prohibitively time consuming if a reasonable number of repeats are to be achieved. We consequently test our methods on a simplified version as a proof-of-concept. This allows choice facets of the real problem to be reflected in a convenient approximation that can be used for rapid prototyping. In particular, uncertainty in the critical cases with respect to continuous parameters such as Mach number and altitude, which tend to be discretised, is of interest.

Variable	**Type**	**Notes**
Case Type	Discrete	Manoeuvre or Gust
Mach Number	Continuous	
Altitude	Continuous	
Aircraft Total Mass	Continuous	
Centre of Gravity Position	Continuous	As a fraction of the chord

Table 7.1: Input variables to the approximate loads process.

Our simplified loads process is defined over 1 discrete and 4 continuous inputs, shown in table 7.1. The two discrete case types considered are a 2.5g turn ("manoeuvre") and a 1-cosine gust ("gust").

Additionally, rather than the full list of IQs, we perform the analysis on a small subset. We retrieve bending and torsion measurements at 6 choice nodes: 3 along the span of the wing, and 3 along the span of the horizontal tailplane, corresponding roughly to regions referred to as "inner" "mid" and "outer" locations on the respective surfaces.

All values in this document are normalised.

7.4 Problem Formalisation

We analyse two closely related problems:

1. **Actively searching for the critical load cases using Bayesian optimisation.** This involves approaching the optimisation described by equation (7.4) directly.

2. **Probabilistic load case downselection.** Given a probabilistic model of the load case space exhibiting uncertainty, we aim to identify candidate regions of the space for further analysis.

Using BO, item 1 concerns efficient construction of an emulator capable of finding potentially critical cases, while item 2 is associated with using this emulator to select potentially critical cases from those considered.

7.4.1 Modelling the Loads Process

In both cases we use robust GP models fit using MCMC as described in chapter 5. To handle the discrete case type variable we use a "piecewise" GP, which in practice simply involves constructing a separate emulator over the 4 continuous parameters for each of the discrete variables. Sharing hyperparameters between the two piecewise GPs is possible though not recommended in this case: the physics responsible for producing the loads are not consistent between case types, and there are not convincing arguments for why any particular input variable should have similar effects on gust loads as it does on manoeuvre loads. This can be verified by comparing distributions of the hyperparameters for the gust and manoeuvre models, which confirms this assertion.

We first approach a simplified problem where we consider 1 IQ at a time. This is significantly less unattractive than the apparent need to repeat the analysis for all IQs might imply: though we only optimise a single IQ at a time, results are returned for all of them each time the loads process is queried, thus still indirectly providing (non-optimal) utility for identifying the critical cases for other IQs. Nevertheless, this is less *efficient* than considering the IQs simultaneously.

7.5 Bayesian Optimisation Procedure

Given an initial sample of data, one constructs an emulator and uses an acquisition function given this emulator to determine the next "most useful" point to query. One then queries the (expensive) target function at this point and augments the available data with this new point and the corresponding response. This procedure is repeated iteratively, usually until a certain number of queries to the target function have been performed, but could also be terminated when the expected utility as predicted by the acquisition function falls below a predetermined threshold.

7.5.1 Case Type Variable

Values of individual IQs between the two case types were discovered to be on different scales to the extent that it was immediately clear from the initial sample if a given IQ was maximised by either the gust case or the manoeuvre case.

The case variable was dropped from the optimisation, with the analysis over the four continuous variables simply repeated for that case type to investigate potential differences behaviour between the manoeuvre and gust case types.

7.5.2 Acquisition Function Optimisation

We take the expectation of the chosen acquisition function (see appendix G) over the posterior distribution of the model hyperparameters. Letting $a(\mathbf{x}|y^*, \theta)$ be an arbitrary acquisition function for a GP emulator with posterior predictive distribution y^* and hyperparameters θ, and assuming we have access to a total of S samples of θ drawn from $p(\theta|\mathcal{D}, \pi_\theta)$ using MCMC, the expected utility of a point, $\mathbf{x}$, as determined by a is:

$$\mathbb{E}[a(\mathbf{x}|\mathcal{GP})]_\theta = \frac{1}{S}\sum_{i=1}^{s} a(\mathbf{x}|y^*, \theta_s) \tag{7.5}$$

The above is taken to be negative in this work, such that the point with the most utility is obtained by minimising equation (7.5) over $\mathcal{X}$. Letting $\hat{\mathbf{x}}$ be the "most useful point" as determined by the acquisition function, one finds $\hat{\mathbf{x}}$ by:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathcal{X}} \mathbb{E}[a(\mathbf{x}|y^*)]_\theta \tag{7.6}$$

which we approach using the *Sequential Linear Least Squares Programming* (SLSQP) algorithm of Kraft [62], as implemented in the python module `Scipy`. Since the objective function described by equation (7.6) is cheap to compute, a reasonably dense preliminary grid search of 200 points is used to locate a suitable initialisation point.

7.5.3 Initial Sample

Due to the simplifications made to the loads process model, we use a particularly sparse initial sample of 7 initial load cases across the 4 continuous load case parameters, which are scaled to accept input on $[0, 1]$. Like the experiments conducted in chapter 6, we sample the hypercube $[0, 1]^4$ using a min-max LHS design [40], which is cheap to compute, offers good space filling properties at this number of dimensions, and is more randomised than quasi-random alternatives such as the Sobol sequence; the latter being important when sensitivity to this initial sample is being assessed.

7.5.4 Acquisition Functions

We test two of the most simple acquisition functions: pure exploration (PX; see appendix G.2.1) and expected improvement (EI; see appendix G.3.1). We initially planned to use a third acquisition function – Hernández-Lobato et al. [61]'s *Predictive Entropy Search* (PES) – though encountered difficulties applying the acquisition function to our problem successfully which compromised the usability of the algorithm. These issues are documented in appendix G.3.2.

7.5.5 Psuedocode

The pseudocode provided by algorithm 2 provides a description of BO as applied to critical load case identification. Since use of a superscripted asterisk for both a predicted quantity and a critical load case is notationally confusing, we use the superscript $\cdot^{\max}$ to denote a critical case in this short section.

7.5.6 Performance Evaluation

We report the absolute regret, defined as the absolute difference between the value at the argmax predicted by the emulator, $\hat{y}^{\max}$, and the value of the true maximum,

Algorithm 2 Bayesian Optimisation For Loads (single IQ)

Given: loads process, $l_{\mathbf{z}}$ returning a specific IQ; initial data, $\mathcal{D}_0 = \{\mathbf{X}, \mathbf{y}\}$; GP prior (choice of mean function, covariance kernel); simulation budget, T; acquisition function $a(\cdot|\cdot)$.
$t = 0$
while $t < T$ **do**
    Create empirical hyperprior π_θ from $\mathcal{D}_t$ ▷ see section 5.4
    Use MCMC to obtain S samples of θ from $p(\theta|\mathcal{D}_t, \pi_\theta)$
    (Optional) predict critical case $\mathbf{x}_t^{\max} = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{E}[p(y^*|\mathbf{x}, \mathcal{D}_t, \pi_\theta)]_\theta$
    Find the most informative case $\mathbf{x}_{\text{new}} = \arg\min_{\mathbf{x}\in\mathcal{X}} \mathbb{E}[a(\mathbf{x}|y^*)]_\theta$
    Query $l_{\mathbf{z}}$ at $\mathbf{x}_{\text{new}}$ to get $y_{\text{new}} = l_{\mathbf{z}}(\mathbf{x}_{\text{new}})$
    Augment data with new points $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \bigcup \{\mathbf{x}_{\text{new}}, y_{\text{new}}\}$
    Increment iteration number: $t = t + 1$
end while
Get final guess of critical case $\mathbf{x}_t^{\max} = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{E}[p(y^*|\mathbf{x}, \mathcal{D}_t, \pi_\theta)]_\theta$
Return: guess critical load case, $\mathbf{x}_t^{\max}$

$y^{\max}$:

$$\text{AbsoluteRegret} = |\hat{y}^{\max} - y^{\max}| \tag{7.7}$$

which is calculated after each iteration.

7.6 Results: Bayesian Optimisation

A low number of total iterations (10, in addition to the initial sample) are used as we simply wish to indicate proof of concept.

Each optimisation is repeated 30 times on randomly generated initial grids. We report the median absolute regret at each iteration, along with bounds corresponding to the minimum and maximum absolute regret, indicating worst-case and best-case performance, respectively.

7.6.1 Wing Bending

BO results for wing bending IQs are shown in figure (7.1). EI can be observed to clearly outperform PX in all cases except for the outer-wing gust-induced bending

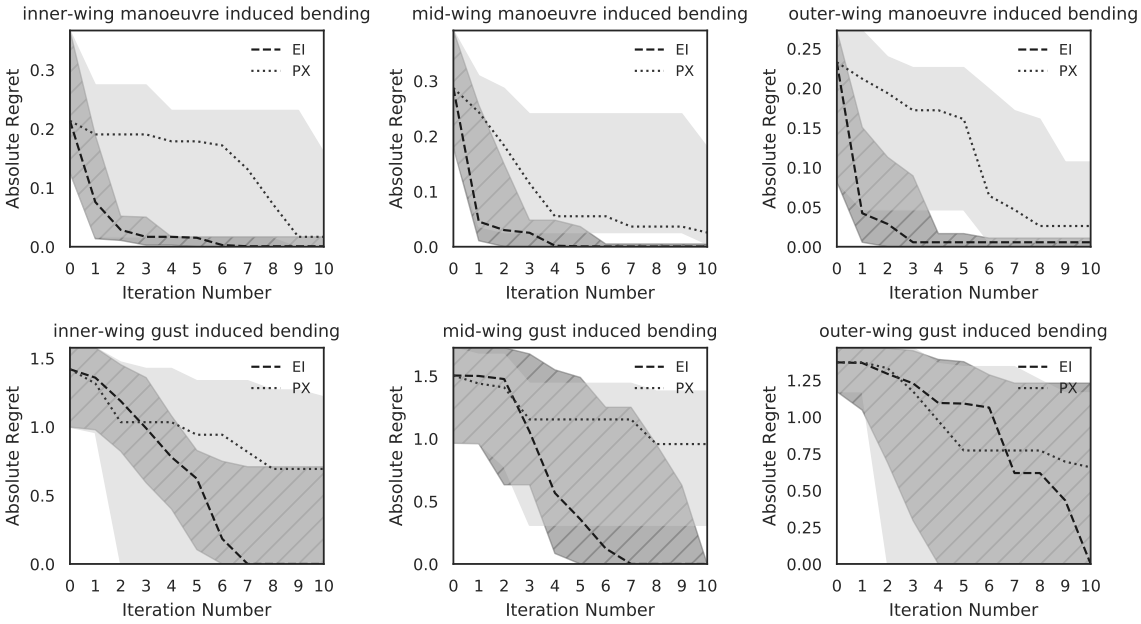

Figure 7.1: Absolute regret for the Bayesian optimisation procedures treating wing bending IQs.

function, on which it performs comparably.

7.6.2 Wing Torsion

Figure (7.2) shows the BO results for wing torsion IQs. EI converges faster and to values closer to the maximum than PX in all cases. Neither EI nor PX converge to the true maximum for mid-wing manoeuvre induced torsion or outer-wing manoeuvre induced torsion within 10 iterations, though the regret is very small (less than 1% of the value at the maximum).

7.6.3 HTP Bending

Results for HTP bending IQs are shown in figure (7.3). EI can be observed to perform either better than or similarly to PX. Regret for the inner-HTP manoeuvre induced bending function appears to exhibit a lot of variation, indicating performance might be sensitive to specifics of the initial sample, though is consistently reduced with more iterations. Regret for the mid and outer HTP manoeuvre-induced bending functions is very small, despite never reaching 0 within 10 iterations.

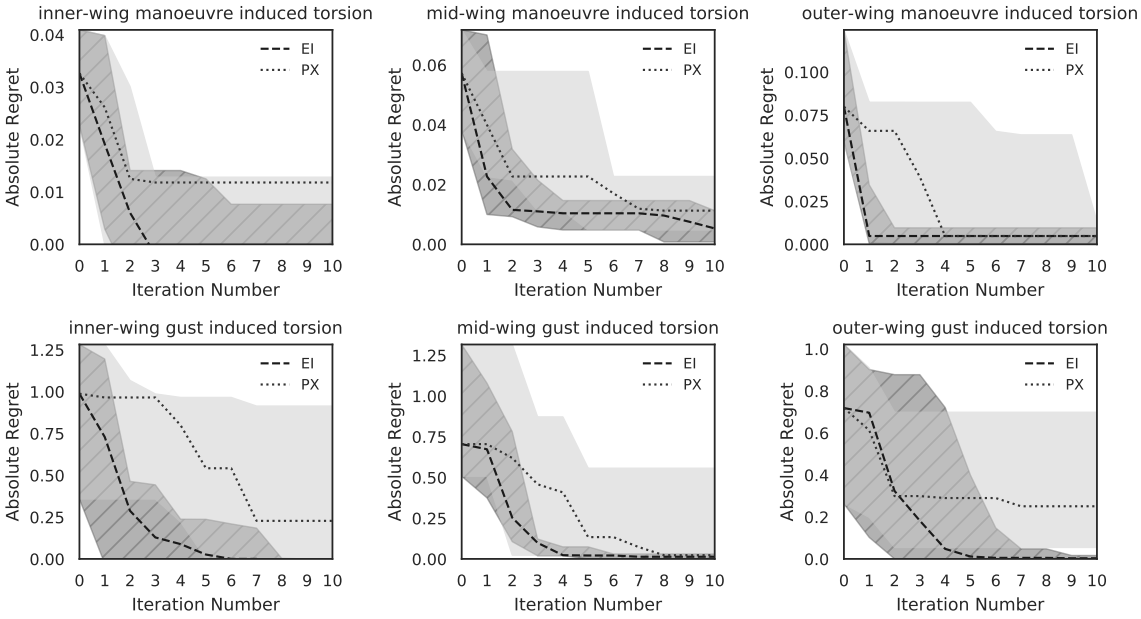

Figure 7.2: Absolute regret for the Bayesian optimisation procedures treating wing torsion IQs.

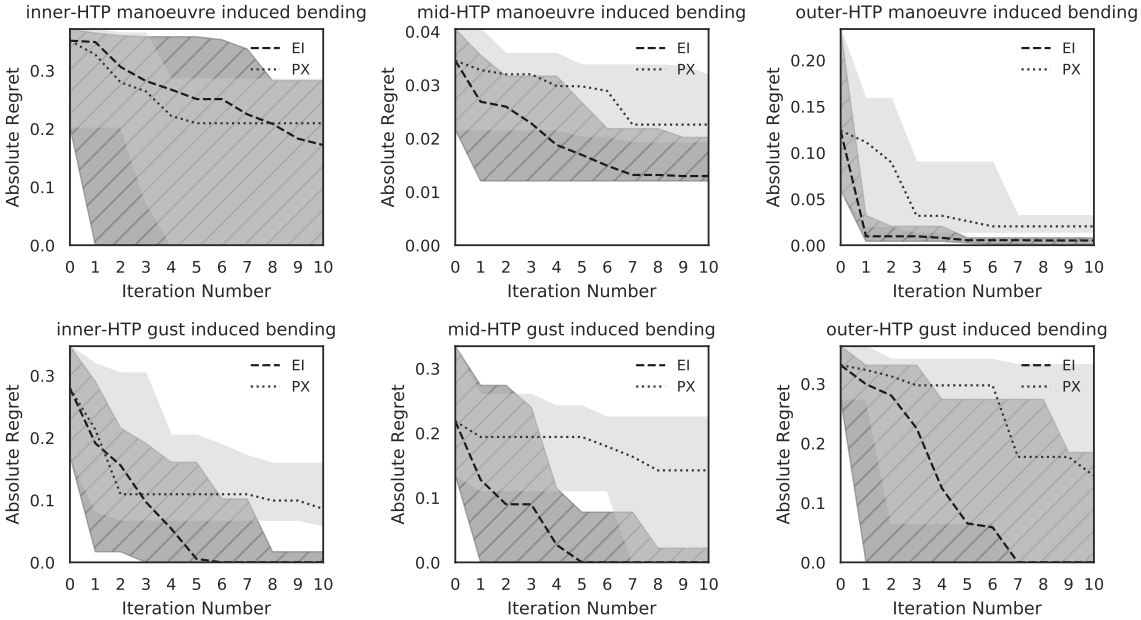

Figure 7.3: Absolute regret for the Bayesian optimisation procedures treating HTP bending IQs.

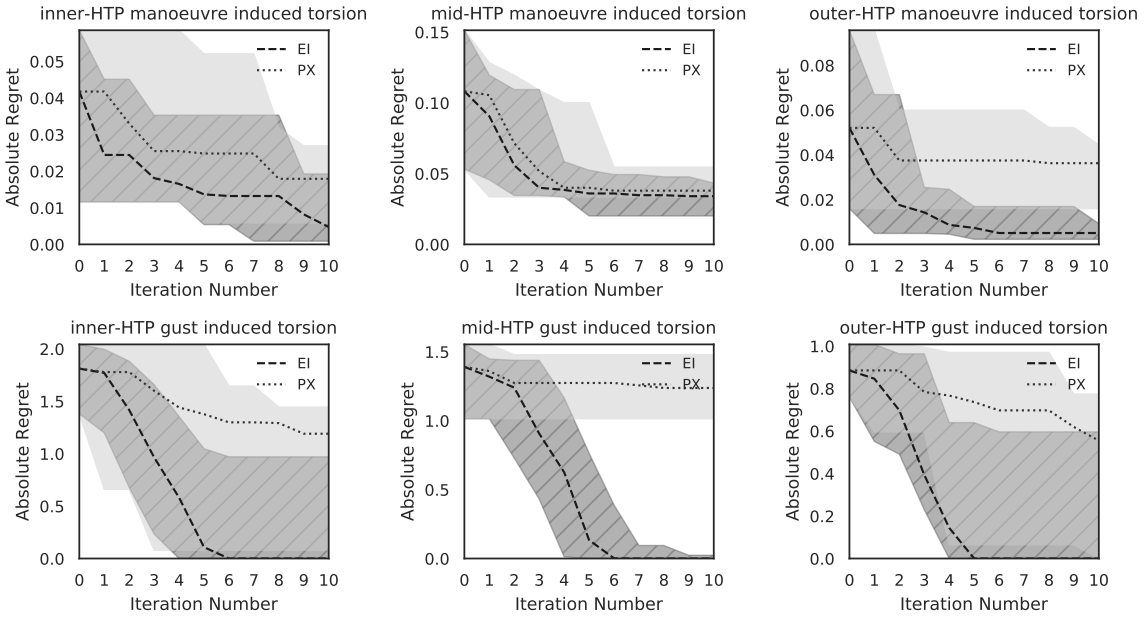

Figure 7.4: Absolute regret for the Bayesian optimisation procedures treating HTP torsion IQs.

7.6.4 HTP Torsion

Figure (7.4) shows the BO results for HTP torsion IQs. EI once again consistently outperforms PX, particularly on the gust-loading functions. Both algorithms struggle to identify the true maximum within 10 iterations for mid-HTP manoeuvre induced torsion, and though the regret is proportionally small, it is slightly larger (approximately 1% of the value at the true maximum) than for other instances of similar behaviour.

7.6.5 Analysis

As a proof of concept, our experiments show that BO is a sample efficient means of exploring the load case space with respect to continuous parameters. Expected Improvement is a simple and effective acquisition function for this purpose, performing more efficiently than simply adding data at the points of greatest predictive uncertainty in terms of ability to identify the maximum of the target function.

7.7 Probabilistic Downselection

Given an emulator for the loads process, we can consider how to use it to estimate the locations of *possibly* critical load cases. More precisely, this involves constructing a distribution over possible locations of the maximum of a function modelled by a Gaussian Process, then using this distribution to restrict further investigation into this "subset" of points. This problem is approached as part of the *Predictive Entropy Search* (PES) [61] algorithm.

Here, we use the same method as PES, and use random features to approximate sample paths from the GP predictive distribution. These randomly generated sample paths permit a cheap, analytical representation which can be optimised quickly, and aggregating the argmaxes of many of these randomised approximations provides a good representation of the distribution over locations for the true argmax. The quality of this approximation is conditional on how well the emulator used to generate the sample paths approximates uncertainty in the target function.

Given this distribution over possibly critical cases, a practitioner can then perform a qualitative assessment on which load cases should be considered further, for example by keeping only those with a certain probability of being critical. This type of analysis offers potentially enormous savings in terms of simulation resources.

7.7.1 Assumptions

The intended method is valid only for stationary GPs (or more precisely, "shift invariant kernels"), due to the way the random feature approximations are generated. For additional details, readers are directed to the paper of Hernández-Lobato et al. [61].

7.7.2 Problem Formalisation

Given an emulator for a particular load measurement, we wish to invoke a distribution over possible locations for the maximum over the load case space, using the random feature approximation method described in [61, appendix A].

Algorithm 3 Approximating $p(\overline{\mathbf{x}}|\mathcal{D}, \pi_\theta)$ using random feature approximations

Given: GP emulator, $\mathcal{GP}$, given data $\mathcal{D}$, S samples of hyperparameters from posterior distribution, $[\theta_1, \ldots, \theta_S]$; requested number of samples, M;
Initialise iteration number: $m = 0$
Initialise array of sampled maxima: $\overline{\mathbf{X}} = \{\}$
while $m < M$ **do**
    Select $\hat{\theta}$ uniformly at random from $[\theta_1, \ldots, \theta_S]$
    Generate a random feature approximation, $\Phi(\mathbf{x}^*)$, of $\mathcal{GP}(\mathbf{x}^*|\mathcal{D}, \hat{\theta})$ ▷ See [61]
    Find $\overline{\mathbf{x}}_m = \arg\max_{\mathbf{x}\in\mathcal{X}} \Phi(\mathbf{x})$
    Add $\overline{\mathbf{x}}_m$ to $\overline{\mathbf{X}}$
    Increment iteration number: $m = m + 1$
end while
Return: $\overline{\mathbf{X}}$

Let $y = l_{\mathbf{z}}(\mathbf{x})$ represent the true loads process for the IQ under consideration, and let $\overline{y} = l_{\mathbf{z}}(\overline{\mathbf{x}})$ represent the maximum absolute load, $\overline{y}$ at the critical load case, $\overline{\mathbf{x}}$. By constructing an emulator for $l_{\mathbf{z}}$, we wish to obtain a probabilistic representation of *possible* locations for the critical case in the form of a probability distribution $p(\overline{\mathbf{x}}^*|\mathcal{D}, \pi_\theta)$.

This distribution is conditional on the hyperprior, π_θ, rather than θ itself, as we marginalise over $p(\theta|\mathcal{D}, \pi_\theta)$. We construct an empirical approximation by sampling a total of M function maximisers from randomly constructed feature approximations to the emulator representing $l_{\mathbf{z}}$ using randomly selected hyperparameters from $p(\theta|\mathcal{D}, \pi_\theta)$. A pseudocode is provided in algorithm 3.

7.7.3 Experiments

We sample approximated function maximisers for three IQS: outer wing bending and torsion during a gust case, and inner wing bending during a turn manoeuvre. Of the 6 combinations of IQs and load cases introduced earlier in this chapter (inner, mid and outer nodes on the wing and HTP, for both bending and torsion), these three are selected due to being the most interesting for designers, with manoeuvre loading being a strong candidate for being critical for inner-wing bending, and gust loading being a strong candidate for being critical for outer wing bending and outer wing torsion.

We use robust GP emulators, constructed as described in chapter (5), by sampling the space of load cases, $\mathcal{X}$, using the Sobol sequence [58] with varying sample sizes $N = [20, 30, 40, 50]$, in order to demonstrate the effect of reducing emulator uncertainty on the ability of the procedure to accurately predict the location of the argmax.

To visualise the resulting 4D distributions, we use *parallel co-ordinates* [63], such that each line on the figure represents a point in 4D space. To communicate a probability distribution on this plot, a single line is configured to feature low transparency, such that darker regions of the plot correspond to higher probability regions.

We also plot the marginal histograms for each input variable across its support, indicating the relative counts for the number of times a sampled maximum falls within that bin.

7.8 Results: Probabilistic Downselection

Distribution over possible critical cases for inner wing bending during manoeuvre loading are shown in figure (7.5) and distributions over possible critical cases for outer wing bending and torsion are shown in figures (7.6) and (7.7), respectively.

In all cases, the number of greyed-out bins on the histograms – which correspond to regions of the load-case space containing less than 1% of the total argmax sample count for that dimension – can be seen to shrink with increasing sample size, as the model becomes more confident in predicting the location of the critical case.

7.8.1 Analysis

The visualisations shown in figures (7.5-7.7) are a potentially invaluable tool for performing qualitative, probabilistic load case downselection, possibly providing large savings in terms of simulation resources if large regions of the search space can be confidently ruled out.

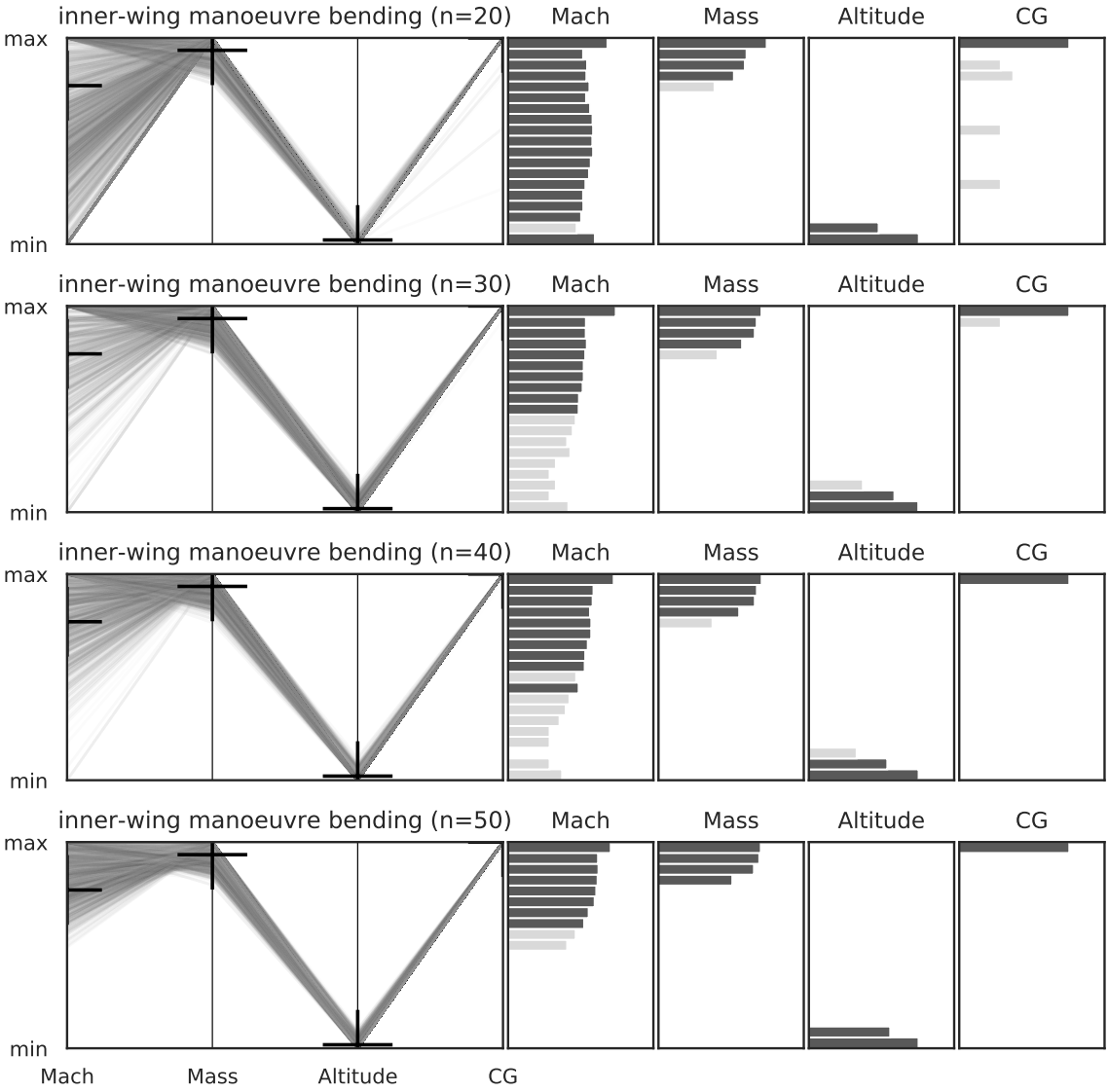

Figure 7.5: Distribution over possible critical cases for inner wing bending during manoeuvre loading as a function of Mach number, total mass, altitude and CG position, for sample sizes $N = [20, 30, 40, 50]$. The left side of each row shows a parallel co-ordinate representation of the 4D probability distribution, with black crosses showing the location of the true critical case. The right side shows the marginal histograms of each input variable, indicating the number of critical case samples occurring within each bin. Bins with less than 1% of the total sample count are shaded out, corresponding to regions of the load case space estimated to having less than a 1% chance of containing the critical case.

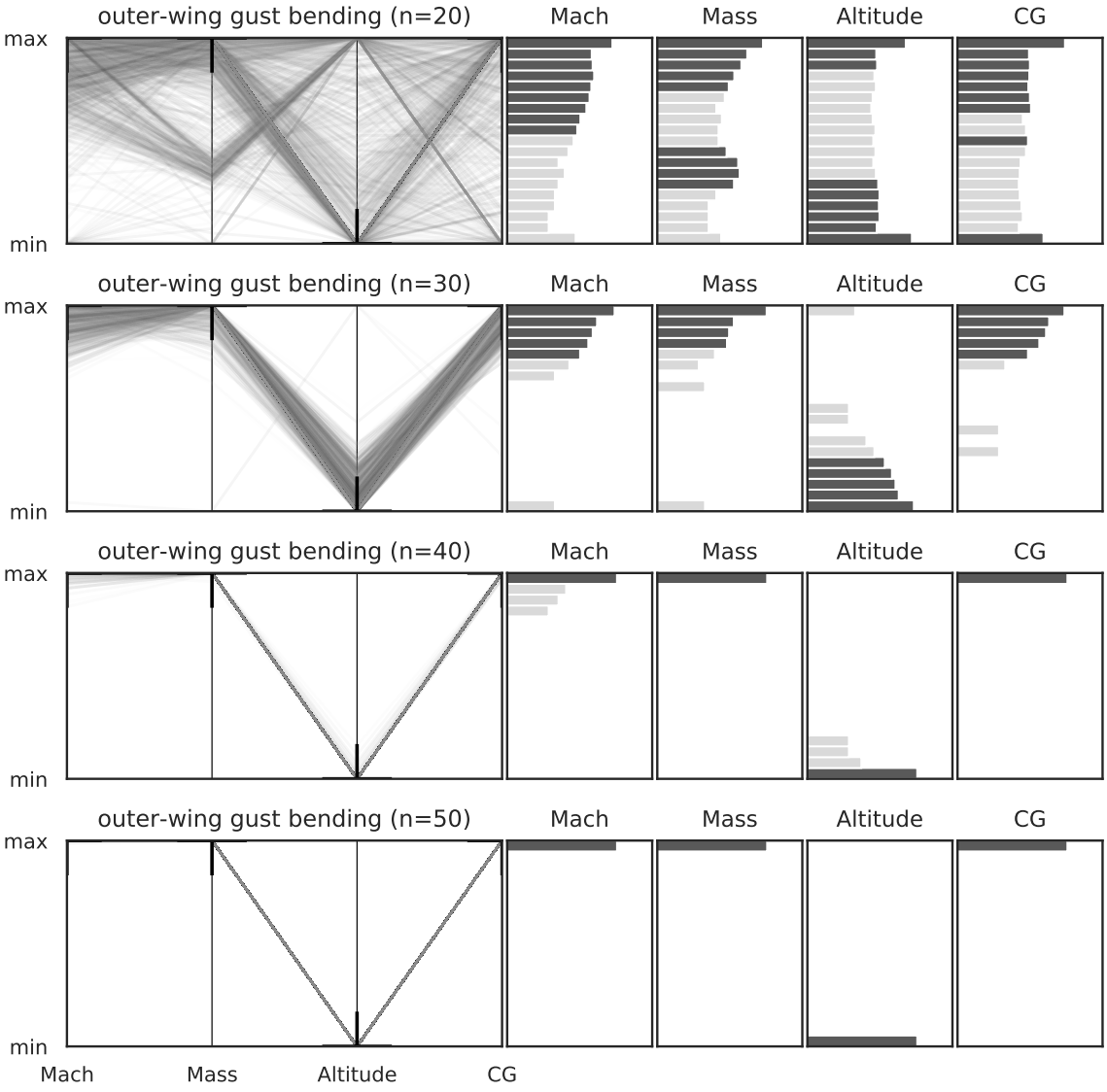

Figure 7.6: Distribution over possible critical cases for outer wing bending during gust loading as a function of Mach number, total mass, altitude and CG position, for sample sizes $N = [20, 30, 40, 50]$. The left side of each row shows a parallel co-ordinate representation of the 4D probability distribution, with black crosses showing the location of the true critical case. The right side shows the marginal histograms of each input variable, indicating the number of critical case samples occurring within each bin. Bins with less than 1% of the total sample count are shaded out, corresponding to regions of the load case space estimated to having less than a 1% chance of containing the critical case.

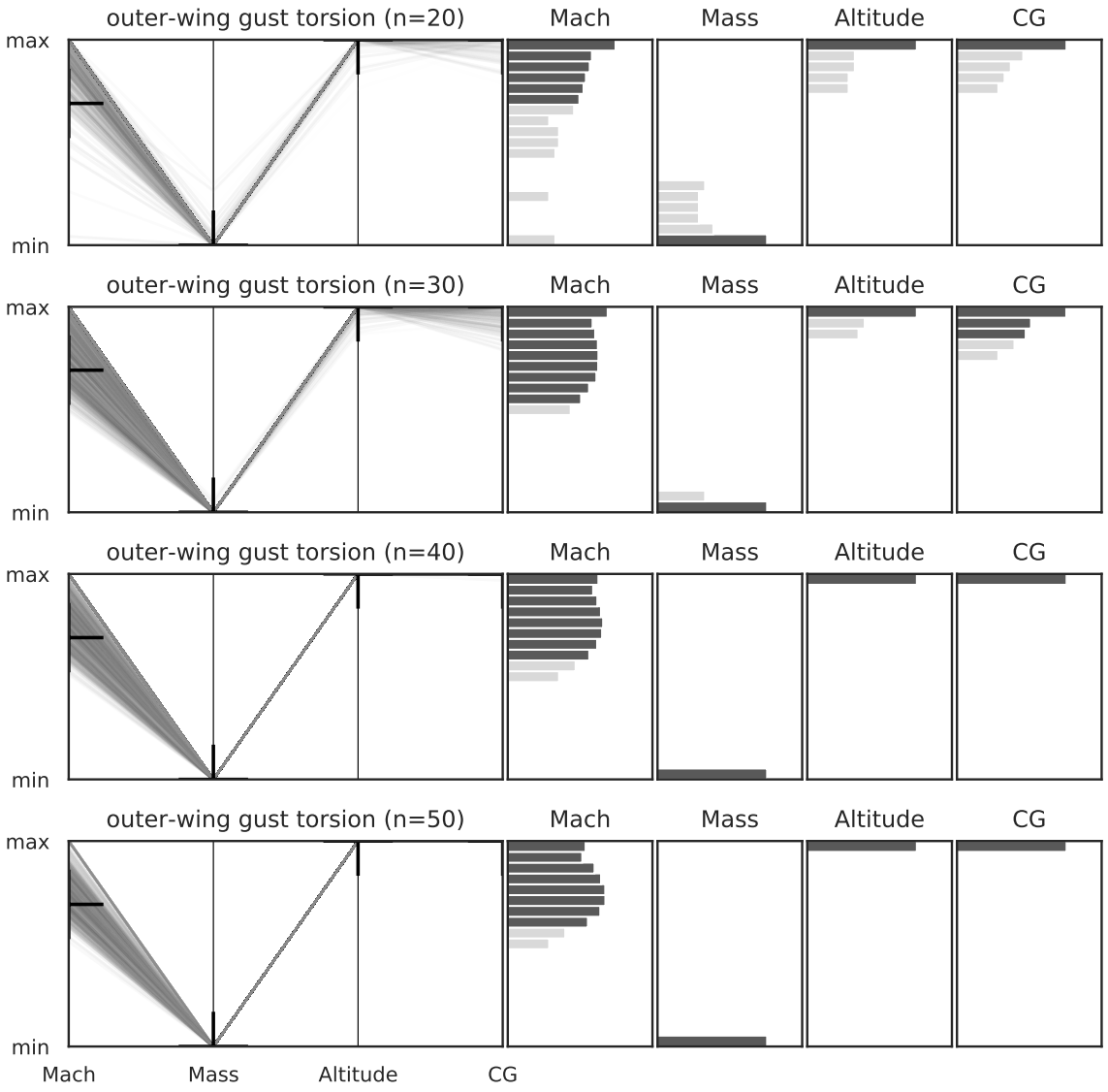

Figure 7.7: Distribution over possible critical cases for outer wing torsion during gust loading as a function of Mach number, total mass, altitude and CG position, for sample sizes $N = [20, 30, 40, 50]$. The left side of each row shows a parallel co-ordinate representation of the 4D probability distribution, with black crosses showing the location of the true critical case. The right side shows the marginal histograms of each input variable, indicating the number of critical case samples occurring within each bin. Bins with less than 1% of the total sample count are shaded out, corresponding to regions of the load case space estimated to having less than a 1% chance of containing the critical case.

In principle, such a tool could also be configured for multiple IQs at once: this is a trivial extension, as one could simply concatenate samples of the critical cases for multiple IQs together an perform a joint analysis, invoking a distribution over locations which potentially maximise *any* of the considered IQs.

7.9 Extensions

A number of extensions can be considered to the critical case search problem.

The Max-value Entropy Search (MES) [64] and Fast Information-Theoretic Bayesian Optimisation (FITBO) [65] algorithms approach similar problems, but are concerned with predicting the *value* of maximum rather than its location: this is potentially as useful if rather than the critical cases one seeks a probabilistic representation of the envelope.

Simultaneous optimisation of more than one (assumed) independent IQs can be considered by a multi-task optimisation framework, as discussed in appendix G.5. This is relatively trivial, though does depend on appropriate scaling of the individual tasks. Accounting for the correlation between IQs is even more realistic, though extending BO to a multiobjective objective function is a challenging problem, particularly for large numbers of objectives. This is discussed in appendix G.6.

Batch acquisition of points is potentially both more effective for finite simulation budgets (see, for example, [66]) *and* more practically suited to industrial use cases than the sequential ("one-at-a-time") acquisition methods used in this chapter, however brings about additional and often complex implementation challenges. These are discussed briefly in appendix G.4.

The PES acquisition function is – in principle – conceptually ideal for the critical load case identification problem, and can be configured to run in multiobjective [67] and/or batch [68] implementations. Unfortunately, both modifications suffer from the same practical limitations as PES (documented in appendix G.3.2), and further research is required if the acquisition function can be stabilised enough for production-grade work.

Extending the method to handle discrete input parameters is another interesting avenue for further work, potentially accomplished using a piecewise Gaussian Process [69].

7.10 Summary

We have introduced the problem of critical load case identification, and discussed a sample-efficient means of approaching it using Bayesian Optimisation. Proof-of-concept experiments on a simplified model indicate the method is performant, and suitable for optimising the loads process over continuous parameters that are usually discretised. Further work could enhance the efficiency of this procedure by considering multiple IQs at once.

We have also investigated a method for analysing uncertainty in the location of the critical cases. Access to this information permits probabilistic load case downselection: a quantitative procedure for reducing the number of load cases one may wish to consider in subsequent analysis by using an explicit model of critical load case uncertainty. This potentially offers huge savings in simulation resources by allowing a practitioner to confidently rule out large regions of the load case space.

Related methods are capable of predicting the *value* of the critical loads, if a probabilistic representation of the envelope is required, rather than the critical cases themselves.

Since both of these methods make use of GP emulators to model uncertainty, they are subject to scaling limitations of up to (approximately) 15 input dimensions, beyond which an prohibitively large amount of data begins to be required for sufficient input-space coverage. Additionally, neither method accounts for uncertainty in the loads due to design variability, which is the subject of the next chapter.

Chapter 8

Design Variability Induced Random Loads

The first stage to approaching the ultimate problem of critical load case identification featuring design variability is a capable model of the loads process which includes the influence of the design vector. In this chapter, we provide an original interpretation of the problem as a stochastic process, and attempt to model it by parametrising the marginal distributions.

8.1 Joint Modelling of the Loads-Design Space

In the previous chapter, the loads process was assumed to receive a fixed design vector in order to make the optimisation problem tractable. This is necessary to navigate difficulties associated with the fact that at the stage of the design process at which the critical load identification problem is considered the design does not have full definition, and one would have to repeat the optimisation each time the design changes, which is expensive.

A conceptually straightforward solution to this problem is to construct a surrogate model for the loads process which takes into account potential design variability. This would, in principle, make repeating the optimisation problem for each change to the design vector tractable by reducing the cost of the objective function.

In simple terms, we would seek to create a surrogate model, $\hat{l}$ usable in place of the real load process, l, on which to perform optimisation over the load case space, $\mathcal{X}$, for different design vectors, $\mathbf{z}$:

$$l(\mathbf{x}, \mathbf{z}) \approx \hat{l}(\mathbf{x}, \mathbf{z}) \tag{8.1}$$

The surrogate, $\hat{l}$, is constructed by sampling the tensor product space, $\mathcal{X} \otimes \mathcal{Z}$, and performing a regression on the input-output pairs for each IQ.

The tensor product space, $\mathcal{X} \otimes \mathcal{Z}$, however, is large – particularly given the cost of data acquisition – to the extent that building an acceptably accurate model over such a space is infeasible due to the curse of dimensionality. Additionally, parametrisation of $\mathbf{z}$ (a "design") in the first place is a highly non-trivial endeavour due to difficulties in parametrisation of fundamentally continuous quantities such as structural stiffness: the design vectors considered in this section, for instance, contain a total of 64 wing stiffness parameters.

Explicit dimension reduction (such as, for example, PCA as applied in chapter 4) over $\mathcal{X} \otimes \mathcal{Z}$ also fails to mitigate this difficulty, as provision of a dataset on which to perform this analysis to a suitable level of accuracy in the first place is also

expensive. The type of design changes under consideration can also have influence over the choice of design parametrisation, meaning investing simulation resources in such a dataset is not necessarily "reusable" for subsequent analyses.

8.2 Loads As a Stochastic Process

Recall that the design vector, $\mathbf{z}$, behaves like a random variable: since we are unable to fix the design, it is permitted to take any value from the set of feasible designs (the design space), $\mathcal{Z}$, with some probability, $p(\mathbf{z})$.

A simple way of reasoning with the distribution of probability over $\mathcal{Z}$ is to assume that each design variable in the vector, $\mathbf{z}$, is independent and has uniform probability over it's support, or in other words, that each possible design change is equally likely. In this case, a simple model for the probability of a design vector of length D, $\mathbf{z} \triangleq [z_1 \dots z_D]$, is as follows:

$$p(\mathbf{z}) \sim \prod_{i=1}^{D} p(z_i), \quad p(z_i) \sim \text{Uniform}(a_i, b_i) \tag{8.2}$$

with a_i and b_i representing dimensionwise lower and upper bounds on each constituent design variable contained in the design vector. We note that more complex (and indeed, realistic) models of $p(\mathbf{z})$ could be introduced by using different marginal distributions and/or introducing correlations between different design variables using a copula. This is approached by Krupa et al. [3].

We summarise the choice of probability distribution for $\mathbf{z}$ by $\pi_{\mathbf{z}}$, and represent the design as a random variable $Z \sim \pi_{\mathbf{z}}$.

Any function of Z is thus stochastic, in the sense that its output must be a probability distribution, too. It follows that the probability of a particular vector of loads $\mathbf{y} = l(\mathbf{x}, \mathbf{z})$, for some nominal load case, $\mathbf{x}$, is equivalent to the probability of that specific design vector, $\mathbf{z}$, given $\pi_{\mathbf{z}}$:

$$p(l(\mathbf{x}, Z = \mathbf{z})|\pi_{\mathbf{z}}) = p(Z = \mathbf{z}|\pi_{\mathbf{z}}) \tag{8.3}$$

By interpreting $\mathbf{x}$ as an index, equation (8.3) can be observed to represent the marginal distribution of a stochastic process defined over $\mathcal{X}$; that is, each different design, $\mathbf{z}$, encodes a "random function", $l(\mathbf{x}|Z)$, defined over $\mathcal{X}$, occurring with probability $p(Z|\pi_{\mathbf{z}})$.

8.3 Probabilistic Loads

Analysing the loads process in this fashion provides several potential benefits.

First, issues associated with the curse of dimensionality are largely obviated, as the design vector, $\mathbf{z}$, is never directly utilised (other than implicitly deciding the joint distribution from which it is generated). This also somewhat relaxes the difficulties associated with parametrisation of $\mathbf{z}$, as one can use a copula to model potential correlations between individual design variables, possibly marginalising the model over different correlation structures if they are themselves uncertain [3].

Second, such a model is still theoretically capable of supplying the information required to progress the design. By identifying load cases with a sufficiently high probability of being critical, one is able to perform a *quantitative* load case downselection, and a corresponding quantification of uncertainty associated with the discretisation of the load case space in the desired manner. Detailed analysis of individual designs may then be performed on this critical subset, perhaps utilising similar techniques as described in chapter 4.

We will create a stochastic process model of the loads process by first reasoning with the marginal distributions of the stochastic process, and then by constructing a model for these marginals as a function of the index.

8.3.1 Notation

We briefly recap some of the potentially confusing notation for this problem, particularly to distinguish between specific, deterministic inputs, and hypothetical, random inputs (that is, inputs that represent "possible" design decisions yet to be made):

- $\mathbf{x}$: vector describing a load case
- $\mathcal{X}$: space of all load cases
- $\mathbf{z}$: a *specific* design vector
- Z : a *random* (uncertain) design vector
- $\mathbf{y}$: a *specific* vector of loads
- Y : a *random* (uncertain) vector of loads
- $l(\mathbf{x}, \mathbf{z})$ loads process (with deterministic inputs)
- $l(\mathbf{x}, Z)$ stochastic loads process: a random function defined over $\mathcal{X}$; accepts $\mathbf{x}$ as an input, and is random due to Z.
- $l(Z|\mathbf{x})$ uncertain output of the loads process at a specific load case, returning a vector of random loads; i.e. $Y = l(Z|\mathbf{x})$.

8.3.2 Marginal Distributions

The marginal distributions of the loads process represent the distribution of loads for a particular load case due to design variability. We denote the marginals at $\mathbf{x}$ by $l(Z|\mathbf{x})$, noting that $Z \sim \pi_{\mathbf{z}}$. Since the output of $l(\mathbf{x}, \mathbf{z})$ is a vector of IQs, $l(Z|\mathbf{x})$ describes a D-dimensional multivariate distribution over a random variable representing the loads, Y:

$$Y = [Y_1 \dots Y_D] \sim l(Z|\mathbf{x}) \tag{8.4}$$

with D being the number of individual IQs, $y \in \mathbf{y} = l(\mathbf{x}, \mathbf{z})$. In other words, equation (8.4) describes a (potentially correlated) joint distribution of loads due to design variability at a particular load case, $\mathbf{x}$, due to variability in the design vector, Z.

8.4 Independent Loads

The most simple model for $l(\mathbf{x}, Z)$ assumes that the marginals are independent. Though this is unrealistic in practice, it permits a model for the marginal distributions without considering any correlation between them, which simplifies the prob-

lem considerably. This is similar to assuming the variation in loads due to design variability at a particular load case behaves as noise.

One way of approaching this is to parametrise the marginal distributions, and then perform a regression on their parameters over the index, $\mathcal{X}$.

Let an arbitrary marginal distribution, $Y = l(Z|\mathbf{x})$, be parametrised by a parametric distribution $h(\vartheta)$. More explicitly, this equates to assuming the probability of the loads at a load case, $\mathbf{x}$, is distributed according to some probability distribution, h, which has parameters ϑ:

$$p(\mathbf{y}|\mathbf{x}, Z) \sim h(\vartheta) \tag{8.5}$$

Since the marginal distributions depend on the position in the index at which they are observed, this implies that the value of ϑ is conditional on $\mathbf{x}$. We can hence consider modelling ϑ as a function of $\mathbf{x}$, thus providing the capability to estimate the marginal distribution of the loads for arbitrary load cases via ϑ, given the parametric assumption(s) encoded by h.

8.4.1 Parametric Assumptions

In addition to the model of the parameters of the marginal distributions across the index, the quality of an (independent) parametric marginals representation depends on the suitability of the choice of parametric marginal distribution.

We choose to use a normal distribution as a first approximation to the parametric form of the marginal IQ distributions. The justification for this assumption proceeds in two stages.

We first assert that the "size" of design changes implied by the design space, $\mathcal{Z}$, are small (i.e., they represent perturbations about some baseline design). For the range of designs considered, such changes should have an approximately linear effect on the loads: this same reasoning is used to justify the use of principal component analysis for linear dimension reduction in chapter 4.

Next, since the design space is of high dimension, we assert that the central limit theorem applies. As a result of a linearisation approximation on the loads process discussed above, it can be represented as a sum of scaled inputs, and the resulting loads – given independent and identically distribution random inputs – should asymptotically approach a normal distribution for a large number of design variables as a result of the central limit theorem: since the random variables in this case are design variables, they are bounded by definition, and a simple model assumes they are all uniformly probable on their support. While we do not offer a rigorous proof of this assertion, it can be validated to an extent via visual inspection, as will be shown shortly.

8.4.2 Probabilistic Model

Let the data be represented by an $N \times M$ matrix, $\mathbf{Y}$, who's N rows correspond to different load cases, $\mathbf{x}$, and M columns correspond to samples from the marginal distribution $l(Z|\mathbf{x})$. In other words, the i^{th} row of $\mathbf{Y}$ contains M samples of y at load case $\mathbf{x}_i$ due to variability in Z.

Our analysis proceeds in two stages: construction of parametric models for each of the N marginal distributions implied by each of $\mathbf{Y}$'s rows, and subsequently regression of the parameters of these distributions over the load case parameters, $\mathbf{x}$, corresponding to each row.

Let Y_i be a length M vector of samples of y, corresponding to be the i^{th} row of $\mathbf{Y}$. We assume that Y_i is normally distributed, and is hence parametrised by a mean, μ_i, and standard deviation, σ_i, i.e. $Y_i \sim \mathrm{N}(\mu_i, \sigma_i)$.

Assuming we use GP regression to model μ_i and σ_i for $i = 1, \ldots, N$ as a function of their corresponding load case parameters $\mathbf{X} = \mathbf{x}_1 \ldots \mathbf{x}_N$, we have a model functionally equivalent to a *heteroscedastic Gaussian Process*: a GP featuring input dependent Gaussian noise. A simple statement of our probabilistic model is as

follows:

$$\begin{aligned} Y_i &\sim \mathrm{N}(\mu_i, \sigma_i) \quad i = 1, \ldots, N \\ \mu &\sim \mathcal{GP}(\mathbf{0}, k_\mu(\mathbf{X}, \mathbf{X}|\theta_\mu)) \\ \sigma &\sim \mathcal{GP}(\mathbf{0}, k_\sigma(\mathbf{X}, \mathbf{X}|\theta_\sigma)) \\ \theta_\mu &\sim \pi_\mu \\ \theta_\sigma &\sim \pi_\sigma \end{aligned} \tag{8.6}$$

where the subscripts μ and σ imply separate hyperpriors, hyperparameters and kernels are considered for the regression models of μ and σ as a function of $\mathbf{X}$: we perform two separate GP regressions over the inferred parameters of a normal distribution fit to each row of the $N \times M$ array, $\mathbf{Y}$.

Since μ and σ are not observed directly (rather they are inferred from a finite sample), they are predicted quantities and thus exhibit uncertainty. We estimate the variance of the posterior distributions for $p(\mu|Y_i)$ and $p(\sigma|Y_i)$ using MCMC first, then fit GPs featuring a noise variance hyperparameter, ε, (see appendix C.2) to the sample mean of these distributions. The noise of the GP models is deterministically designated to be the median standard deviation of the standard deviations calculated for the N posterior distributions, i.e.:

$$\begin{aligned} \varepsilon_\mu &= \mathrm{Md}(\mathbb{V}[\mu_1|Y_1], \ldots, \mathbb{V}[\mu_N|Y_N]) \\ \varepsilon_\sigma &= \mathrm{Md}(\mathbb{V}[\sigma_N|Y_1], \ldots, \mathbb{V}[\sigma_N|Y_N]) \end{aligned} \tag{8.7}$$

where $\mathbb{V}[\cdot|\cdot]$ represents the sample variance of the argument. We then perform GP regression with deterministic noise on the expectations of the parameter distributions (equivalent to their sample means), with noise variances set to ε_μ and ε_σ for the respectively subscripted model.

$$\begin{aligned} \mu &= [\mathbb{E}[\mu_1|Y_1], \ldots, \mathbb{E}[\mu_N|Y_N]] \\ \sigma &= [\mathbb{E}[\sigma_1|Y_1], \ldots, \mathbb{E}[\sigma_N|Y_N]] \end{aligned} \tag{8.8}$$

Inference of θ for μ and σ can thus be performed separately:

$$\begin{aligned} \mu &\sim \mathcal{GP}(\mathbf{0}, k_\mu(\mathbf{X}, \mathbf{X}|\theta_\mu, \varepsilon_\mu)) \\ \theta_\mu &\sim \pi_\mu \end{aligned} \tag{8.9}$$

$$\begin{aligned} \sigma &\sim \mathcal{GP}(\mathbf{0}, k_\sigma(\mathbf{X}, \mathbf{X}|\theta_\sigma, \varepsilon_\sigma)) \\ \theta_\sigma &\sim \pi_\sigma \end{aligned} \tag{8.10}$$

where the hyperpriors π_μ and π_σ are designated as described in chapter 5, and the length N vectors μ and σ are calculated as described by equation (8.8).

8.4.3 Predictions

In the interest of clarity, we restate that a superscripted asterisk is used to denote a predicted quantity, except in the case of $\mathbf{x}$, where it represents the point in the load case space at which a prediction is made.

For a load case of interest, $\mathbf{x}^*$, we predict the parameters of the corresponding marginal distribution of loads, $\mathrm{N}(\mu^*, \sigma^*)$, via the predictive distributions of our GP models: $p(\mu^*|\mathbf{x}^*, \mathcal{D})$ and $p(\sigma^*|\mathbf{x}^*, \mathcal{D})$, as described in chapter 2.

Point predictions are possible by taking expectations, though this ignores uncertainty in both prediction of the parameters using a GP *and* uncertainty associated with inference of the parameters in the first place. Probabilistic predictions of a single IQ, y^*, should consequently be integrated over the predictive distributions of μ^* and σ^*. Dropping the data argument in the predictive distributions for μ and σ for legibility, we have:

$$p(y^*|\mathbf{x}^*) = \iint p(y^*|\mu^*, \sigma^*)\, p(\mu^*|\mathbf{x}^*)\, p(\sigma^*|\mathbf{x}^*)\, d\mu\, d\sigma \tag{8.11}$$

In practice, this corresponds to predicting y^* using a mixture of normal distributions. The above integration can be performed numerically by sampling μ^* and σ^* from their predictive distributions, thus constructing a Monte-Carlo estimate of

$p(y^*|\mu^*, \sigma^*)$. Assuming S samples of μ^* and σ^* are available, we can approximate the distribution of y^* as follows:

$$p(y^*|\mathbf{x}^*) \approx \frac{1}{S}\sum_{i=1}^{S} p(y^*|\mu_i^*, \sigma_i^*) \tag{8.12}$$

In other words, a prediction consists of a weighted sum of many different normal distributions.

8.4.4 Data

With reference to the previous section, we observe an $N \times M$ matrix of loads, $\mathbf{Y}$. Noting that the loads process is expensive, neither N nor M are usually particularly large. In the following experiments we have access to $M = 60$ different realisations of the stochastic loads process $l(\mathbf{x}, Z)$.

Each realisation is selected at random from a 64-dimensional design space, $\mathcal{Z}$, consisting of different bending and torsional wing stiffness parameters:

$$Z = [Z_1, \ldots, Z_{64}] \quad Z_i \sim \text{Uniform}(\min_i, \max_i), \quad i = 1, \ldots, 64$$

where min and max represent dimensionwise lower and upper bounds on that particular design variable. This distribution may not be realistic in practice, as design variables tend not to vary independently, though modelling this dependency is a challenge.

When sampling the load case space, we set $N = 50$ and use the Sobol sequence to sample the continuous parameters, such that the resulting array of loads, $\mathbf{Y}$, is of size 50×60, and corresponds to an array of load cases, $\mathbf{X}$, of size 50×4.

Visualisations of the data for two load case types – inner wing manoeuvre induced bending and outer wing gust induced bending – are shown in figures (8.1) and (8.2). These case types and IQs are selected due to being highly relevant for the design of the inner and outer wing structure, respectively. Additional (arbitrary) case

types could be considered using this framework, but are not analysed here in these proof-of-concept experiments in the interest of brevity.

Figure 8.1 shows the loads for inner wing manoeuvre induced bending at 50 random combinations of Mach, mass, altitude and CG position as a fraction of the chord. The assumption of normally distributed marginals appearing to be reasonable visually, though it is difficult to assess given the sparsity of the data. Figure 8.2 shows the loads for outer wing gust induced bending at another 50 points in the load case space. The assumption of normally distributed marginals is less effective than for the manoeuvre induced loads in figure (8.1), but is regardless acceptable for most of the load cases.

8.5 Experiments

We construct models for $\mathbf{Y}$ as described in section (8.4.2). Predictions of the distributions of load for that particular IQ are then made at 50 new load cases, $\mathbf{X}^*$.

8.5.1 Results

Figure (8.3) shows the predicted stochastic loads for inner wing manoeuvre-induced bending at 50 new random combinations of Mach, mass, altitude and CG. Each subplot represents a single marginal distribution. We show many superimposed predictions with low transparency, such that darker regions correspond to higher probability. While many of the predicted densities are *acceptable*, some are clearly overly diffuse, represented by the data histograms being barely visible on the plot. The means of the marginal distributions are in general well predicted, which can be confirmed by inspection of figure 8.5.

Figure (8.4) shows the predicted stochastic loads for outer wing gust-induced bending at 50 new random combinations of Mach, mass, altitude and CG. Identically to figure (8.3), each subplot represents a single marginal distribution, and we show many superimposed predictions with low transparency. The model can be seen to perform poorly for this case type, as the validation data histograms are not visible

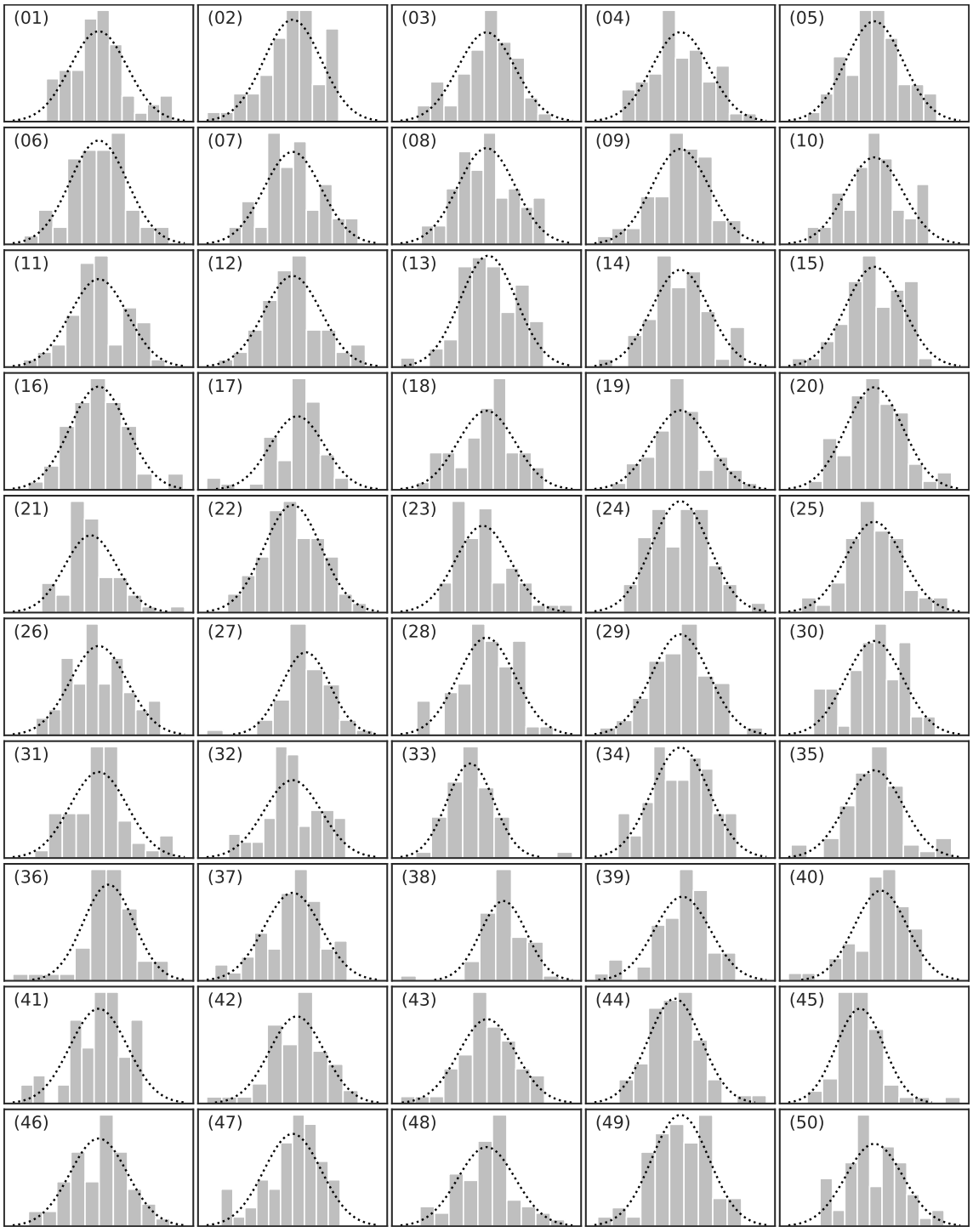

Figure 8.1: Stochastic loads for inner wing manoeuvre induced bending at 50 random combinations of Mach, mass, altitude and CG. Each subplot represents a single marginal distribution. Maximum likelihood normal distributions are also shown to visually demonstrate the parametric assumption discussed in section (8.4.1).

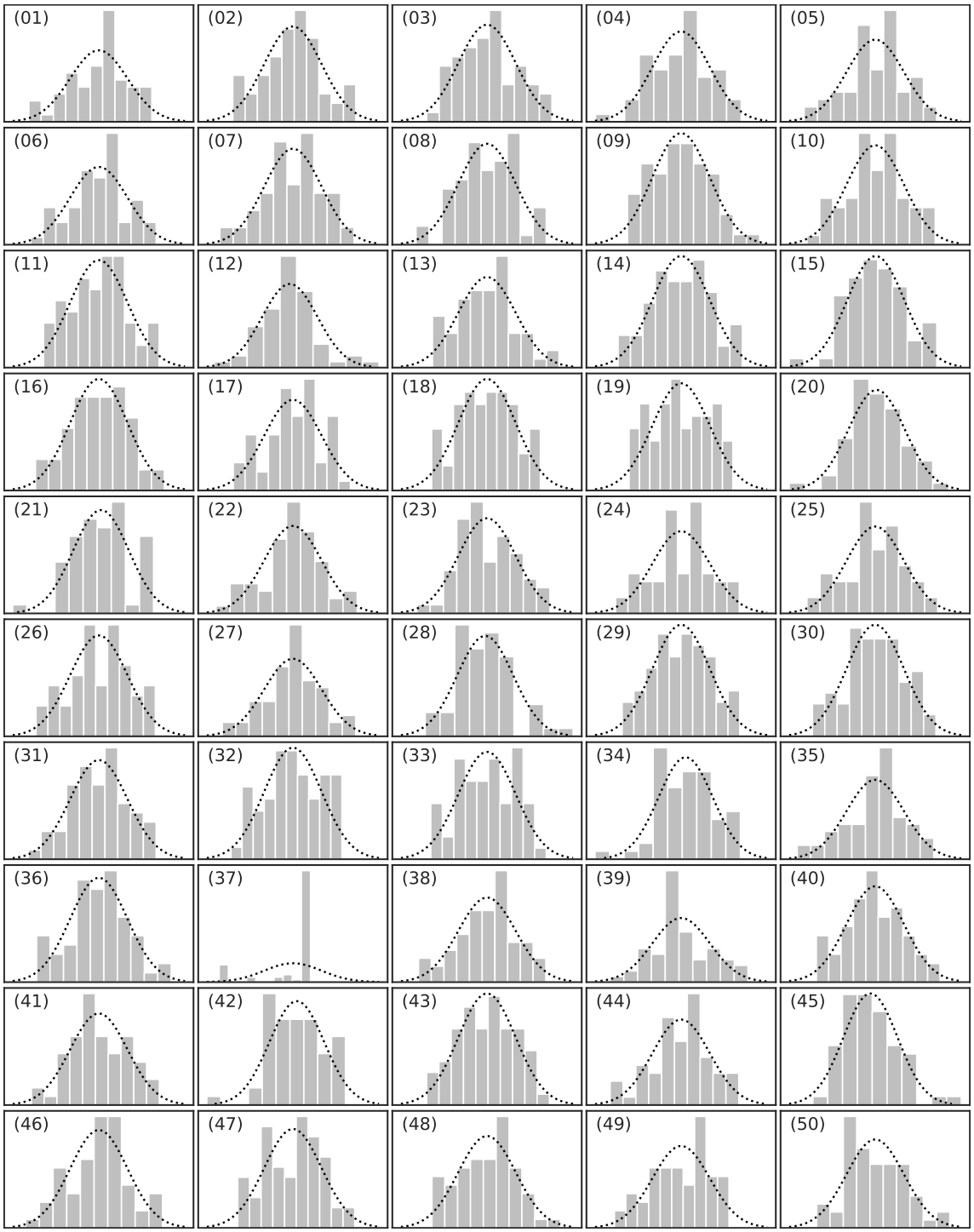

Figure 8.2: Stochastic loads for outer wing gust induced bending at 50 random combinations of Mach, mass, altitude and CG. Each subplot represents a single marginal distribution. Maximum likelihood normal distributions are also shown to visually demonstrate the parametric assumption discussed in section (8.4.1).

at the scale of the plots required to resolve the overly diffuse predictive distributions, and low individual probability means many of these are not visible, either.

Figure (8.5) shows correlation plots of data-inferred and predicted parameters for the marginal distributions of the test data for the manoeuvre case. Errorbars on both axes correspond to 0.05 and 0.95 quantiles for that quantity: the vertical errorbars indicate the predictive uncertainty of the model, while the horizontal errorbars indicate uncertainty in the inferred value of the predicted parameter given the data. The predictions appear to align well for the μ model, indicating the means of the marginal distributions are relatively well predicted. Predictions align slightly less well for the σ model, though most of the incorrect predictions for the latter fall well within reasonable bounds of uncertainty for either prediction (vertical) or inference from finite data (horizontal).

Figure (8.6) shows correlation plots of data-inferred and predicted parameters for the marginal distributions of the test data for the gust case. The marginal distribution means are well predicted (left), with almost all of the data-inferred values for μ falling within reasonable bounds of predictive uncertainty (vertical errorbars) designated by the predictive model. Since there is a serious outlier for which the predicted standard deviation (right) is significantly below the data-inferred value, a zoomed in view of the bottom left of the marginal standard deviation correlation subplot is provided in figure (8.7). Most of the predictions for σ are inaccurate, though do exhibit large amounts of predictive uncertainty.

Kernel lengthscales for the GPs predicting the marginal distribution parameters are shown for both the manoeuvre and gust cases in figure (8.8). The darker histogram represents the manoeuvre case. The plots show the empirical log density of the kernel lengthscales for each dimension, and is cut at 2: since we are interested in the "more nonlinear" behaviour at short (< 1) lengthscales, the asymptotically linear behaviour beyond a lengthscale of approximately 2 is not of interest. The gust case can be seen to behave more nonlinearly with respect to Mach and CG in the case of the μ models, and almost universally more nonlinearly in the case of the σ models. A greater amount of predictive uncertainty can thus be anticipated at points away

from the training data in the gust case than in the manoeuvre case.

8.6 Analysis

Gauging the usefulness of these models is difficult, as there is no currently-implemented method of approaching the problem they seek to solve.

Regardless, it is difficult to recommend use of these models in their current state: while the manoeuvre-induced inner wing bending appears reasonably well represented both visually, as can be seen in figure (8.3), and in terms of predicted parameters, as in figure (8.5), there are still several predicted distributions which are extremely diffuse (specifically, predictions indexed (8), (29), (40) and (47) on figure (8.3)).

The outer wing gust-induced bending distributions are even more difficult to model, with the majority of the predicted distributions exhibiting far too much variance to be useful in any way, as can be observed on figure (8.4). With reference to figure (8.6), this appears to be entirely due to difficulties in predicting the marginal standard deviations, as the means of the marginal distributions are relatively well modelled.

One interpretation of the differences between the results for the two considered cases is that outer-wing gust loading may behave more nonlinearly with respect to the continuous load case parameters than inner-wing manoeuvre loading. This can be confirmed by inspecting the kernel lengthscales for the models of the marginal distribution parameters, shown in figure (8.8). The kernel lengthscales for the gust model are almost universally smaller than those for the manoeuvre model, particularly with respect to Mach number and CG, suggesting the behaviour of the loads with respect to these parameters is more complex for the gust cases, and thus more difficult to model. This is also reflected in figures (8.6) and (8.7) to an extent, too, since large amounts of predictive uncertainty when using a GP model is indicative of predictions being made "far away" from training data in terms of kernel-lengthscale adjusted distance.

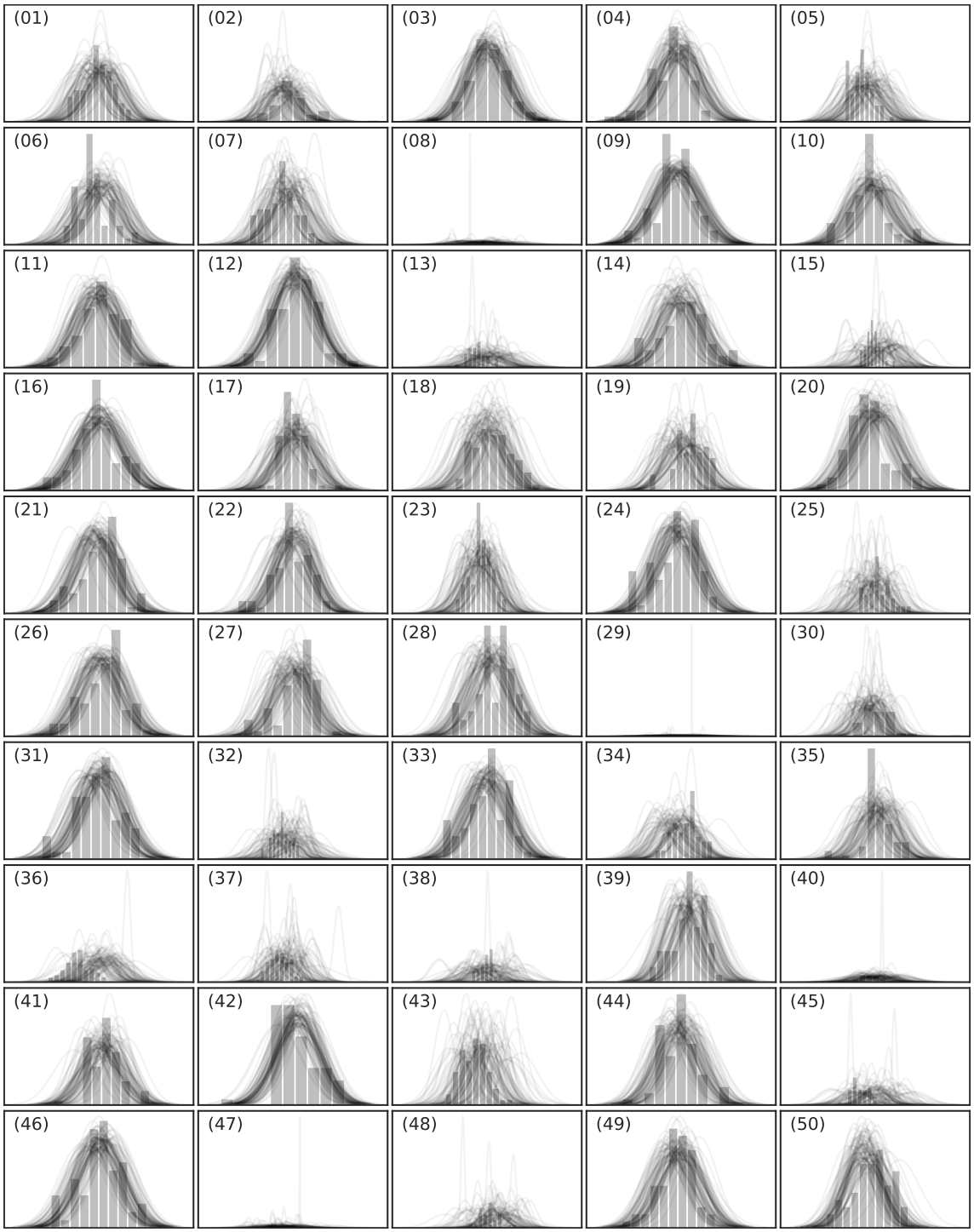

Figure 8.3: Predicted stochastic loads for inner wing manoeuvre-induced bending at 50 new random combinations of Mach, mass, altitude and CG. Each subplot represents a single marginal distribution. We show many superimposed predictions with low transparency, such that darker regions correspond to higher probability.

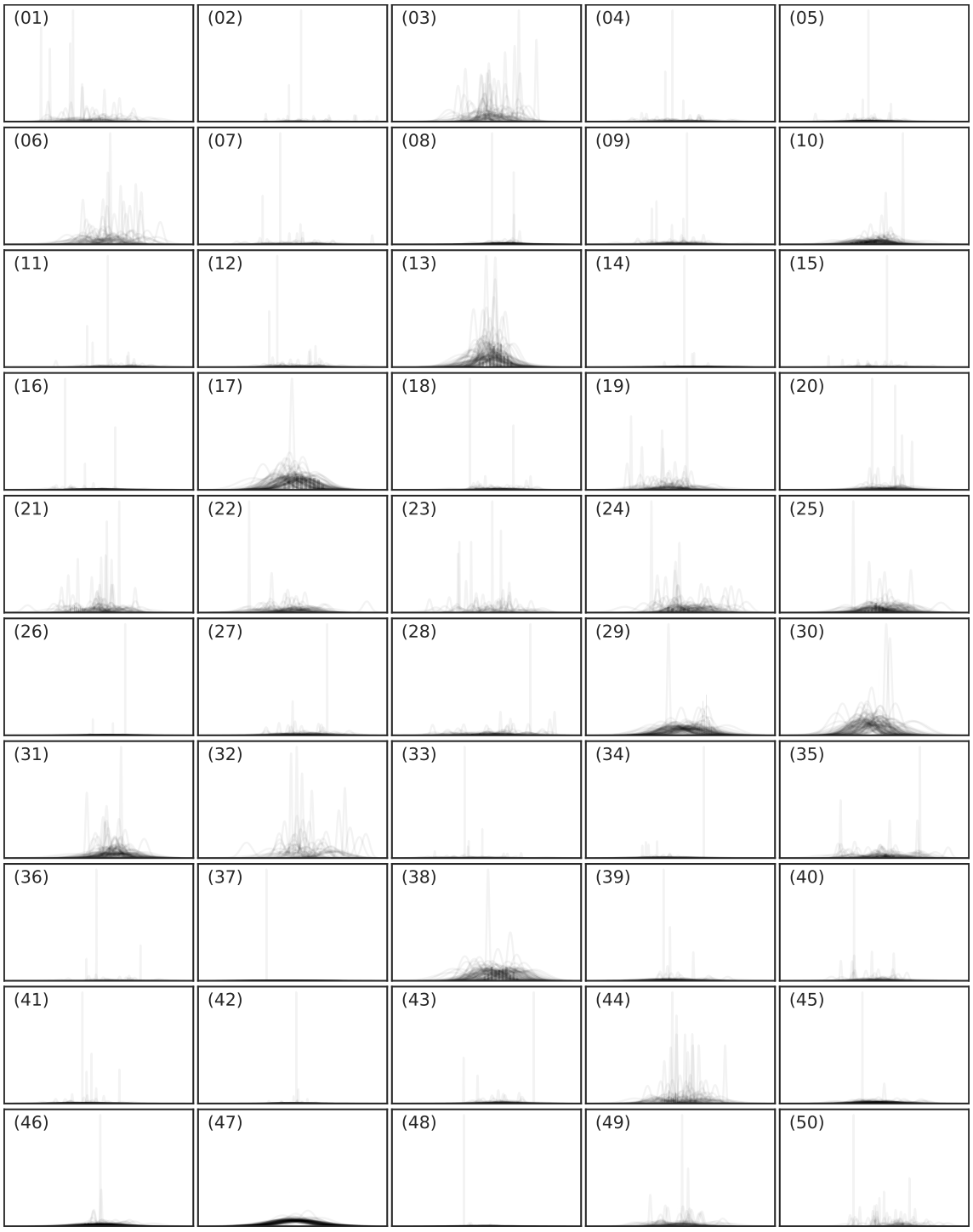

Figure 8.4: Predicted stochastic loads for outer wing gust-induced bending at 50 new random combinations of Mach, mass, altitude and CG. Each subplot represents a single marginal distribution. We show many superimposed predictions with low transparency, such that darker regions correspond to higher probability.

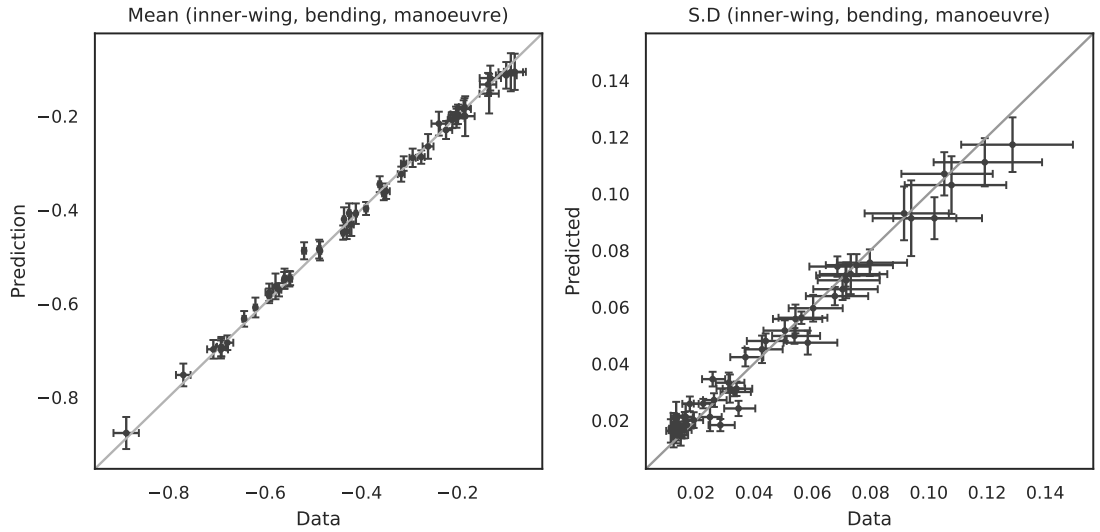

Figure 8.5: Predicted and data-inferred marginal distribution parameters (μ, σ) for the test data for manoeuvre-induced inner wing bending. The errorbars on both axes correspond to the 0.05 and 0.95 quantiles for that quantity, with the central point corresponding to the mean.

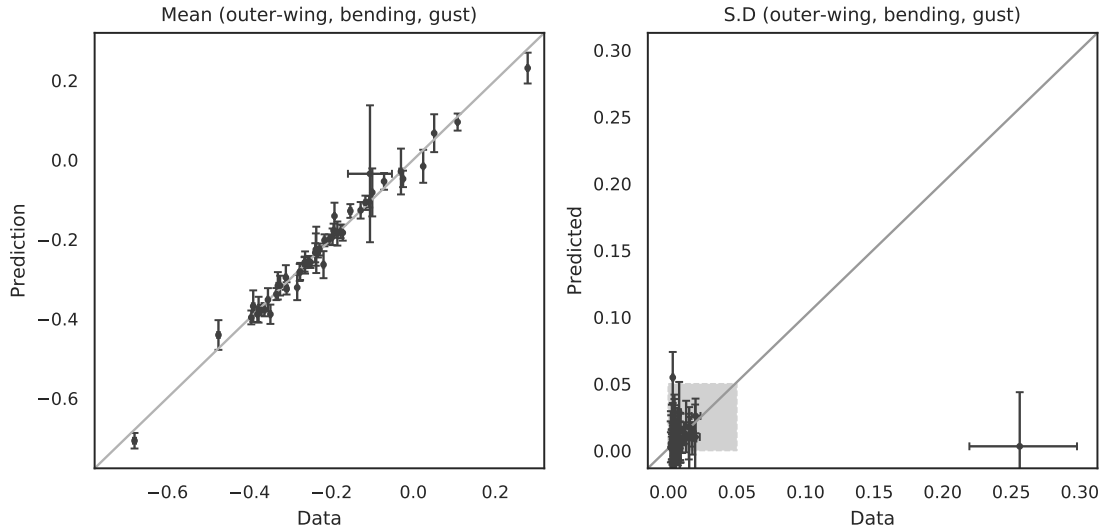

Figure 8.6: Predicted and data-inferred marginal distribution parameters (μ, σ) for the test data for gust-induced outer wing bending. The errorbars on both axes correspond to the 0.05 and 0.95 quantiles for that quantity, with the central point corresponding to the mean. The shaded area on the plot for σ (right) is shown up close in figure 8.7.

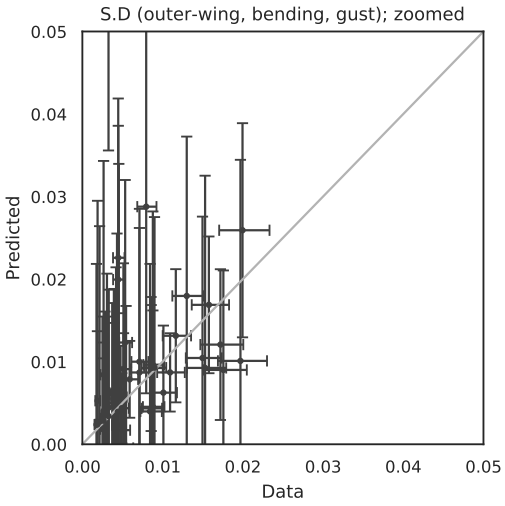

Figure 8.7: Zoomed-in predicted and data-inferred marginal standard deviation (σ) for gust-induced outer wing bending. The plot shows a zoomed-in picture of the shaded area on figure 8.6.

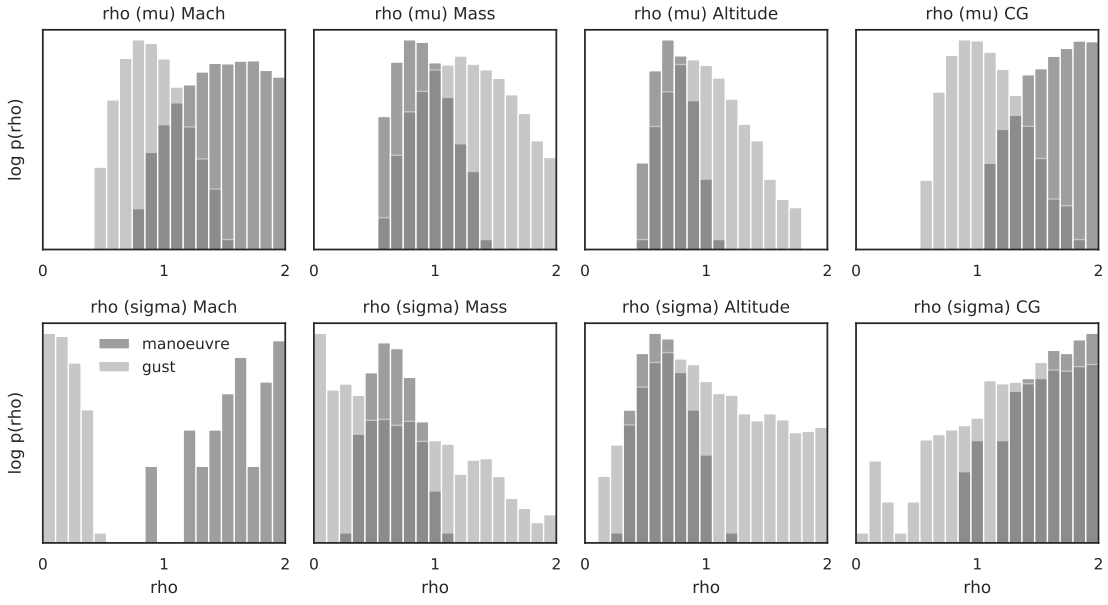

Figure 8.8: Anisotropic lengthscale distributions for the GPs modelling μ and σ for both gust-induced outer wing bending and manoeuvre-induced inner wing bending. The darker histogram represents the manoeuvre case. The plots show the empirical log density of the kernel lengthscales for each dimension.

It is also possible that presence of the outlier visible on figure (8.6), combined with the relatively large amounts of predictive uncertainty visible on both figures (8.6) and (8.7) indicates that the assumptions used to construct the predictive model for this marginal distribution parameter may be unsuitable. The most likely candidate for this is an underestimation in the amount of statistical uncertainty present in inferring the marginal standard deviation from a finite amount of samples, which manifests as sample noise as far as the predictive model for σ is concerned. Such "unrecognised" noise has the potential to introduce overfitting in the kernel lenghtscales (manifesting as a bias in the lengthscales towards zero, which is visible on figure (8.8) for the Mach, Mass and CG axes), since the model is forced to accommodate sharp corners in one or more dimensions that might not be present in a noiseless signal.

The assumption of normally distributed loads is *acceptable* as a first approximation, as seen on figures (8.1) and (8.2). More realistic models of design variability have the potential to invalidate this assumption, due to introducing correlations between design variables, thus implying the central limit theorem does not apply.

Finally, utilising such probabilistic predictions is not straightforward. While one could imagine using the models to assess the probability that loads above a particular threshold are observed for that load case, computing this probability requires sampling, and is thus not as convenient as using the Expected Improvement acquisition function as in the previous chapter. Approximate locations of the critical cases also cannot be sampled as in chapter (7), since the stationary condition required for the method of Hernández-Lobato et al. [61] is violated.

8.7 Further Work

Two primary directions for further work in modelling design variability as an uncertainty are immediately recognisable: modelling of correlations between individual IQs, and relaxing the parametric assumption on the distributions of the marginals.

8.7.1 Correlated IQs

Modelling correlations between IQs is of interest, since individual IQs are almost surely correlated in reality: one can easily imagine that force measurements at two adjacent nodes on the same wing ought to change in similar ways as the conditions responsible for producing such forces are changed. Similarly, quantities which are obtained as functions of such forces – such as moments – will exhibit at least some degree of correlation, too.

Modelling multiple IQs at once also has practical benefits, as access to a single model capable of producing predictions for all (or, at least, multiple) IQs simultaneously is more convenient for practitioners.

We can model correlation between the IQs *implicitly* by supposing the loads are a function of a latent variable, u; with which we compute the Kronecker product with some (scaled) correlation matrix, Σ_y, which designates the correlation between each of the marginal IQs.

Assuming there are R IQs of interest:

$$Y = [Y_1 \dots Y_R] \sim (\text{diag}(\alpha) \diamond \Sigma_y) \otimes u \tag{8.13}$$

where $\text{diag}(\mathbf{a})$ is an $R \times R$ diagonal matrix, whose leading diagonal, $\mathbf{a}$, is a length R vector of scaling factors; $\otimes$ represents a Kronecker product; and $\diamond$ an elementwise product.

Modelling the correlations between the marginal IQs in such a way is equivalent to assuming the *same* (unobserved) physical process is responsible for the changes in the IQs as the load case parameters are changed. This is a more palatable assumption than independence between the IQs, but still requires one to assume a parametric distribution for the correlated distributions.

Since the correlation matrix, Σ_y, is unknown in practice, either the matrix itself or (perhaps more appropriately) its Cholesky factor must be inferred from data. Σ_y consequently behaves as a matrix-valued hyperparameter, and an effective hyper-

prior in this instance is the vine-based method of Lewandowski et al. [70] (sometimes the termed the "LKJ" prior).

8.7.2 Non-Parametric Marginal Distributions

One way of relaxing the parametric assumption on the marginal IQs is by modelling the influence of the design on the loads as a latent *covariate* (input variable) introduced as an additional (fictitious) input variable to a GP defined over the load case space, $\mathcal{X}$. We stress that this latent covariate formulation is different to that in the preceding section, and is equivalent to performing nonlinear dimension reduction over the design space, $\mathcal{Z}$, using a GP [71]. Latent covariate GP regression for the purposes of modelling non-parametric, non-stationary noise is described in detail by Wang and Neal [72].

Such a latent covariate regression model assumes that the variation observed between samples of the loads for each load case is due to presence of unobserved variables (one for each "random design"), interpreted in context to summarise the "missing information" associated with not considering the design vector explicitly. This extra variable introduces additional variability to the model without assuming it has a particular *type*, as was implied by use of a parametrised marginal distributions earlier. Such a formulation naturally models correlation between the marginal load distributions as a function of the load case parameters, too. This is useful since the way in which derivative changes to the design influence the loads is expected to be similar for similar load cases: changing the altitude at which a manoeuvre is performed, for example, ought not to radically change the physics of the resulting load case.

It follows that such techniques are a promising avenue for further investigation in modelling the influence of design variability on the loads process.

8.8 Summary

In this chapter, we consider the loads process featuring structural design variability, and model it as a stochastic process. We propose a simple model for random structural design variability, and via some assumptions, use a normal distribution to parametrise the resulting marginal distributions of the stochastic loads process. Assuming these distributions are independent for simplicity, we attempt to model the parameters over the index using GP regression with mixed success: while the means of the distributions can be relatively well predicted, the standard deviations tend to be overly diffuse, leading to predictions exhibiting very high variance. This is particularly severe for gust-induced outer wing bending loads, which investigation suggests may be due to relatively increased nonlinearity with respect to Mach number when compared to a manoeuvre-induced inner wing bending case, as indicated by the kernel lengthscales of the model.

Such models are consequently of questionable utility in an industrial context in their current state, but are a potential starting point for further investigation.

Chapter 9

Conclusions

9.1 Summary

As set out in chapter 1, this work set out with 4 major objectives, which we restate below.

1. A demonstration of the use of surrogate models for providing rapid approximations of the aircraft loads analysis process, with examples of industrially relevant use-cases utilising such models.

2. A critical assessment of the uncertainties present at the loads-structural coupling interface, investigating the consequences of the use of an expensive, detailed model early in the design process at stages where the structure does not have detailed definition.

3. Development of a models capable of representing the uncertainties described by item 2.

4. A robust, industrialisable method for constructing models of uncertainty, to facilitate the above.

Each objective will be addressed in turn:

9.1.1 Objective 1: Surrogate Models For Loads

Chapter 4 documented the development of surrogate models for the loads process given a discrete selection of pre-selected critical cases. These models are capable of representing the real loads process to a high degree of accuracy for the IQs considered. Example applications using similar models to feed back changes to the structural optimiser are documented in [2], and to propagate correlated design uncertainties in [3].

9.1.2 Objective 2: Assessing Loads-Structural Coupling Uncertainties

At the end of chapter 4 we introduced problems associated with discretisation and downselection of the load case space. Due to the cost associated with querying the

loads process, critical case identification is not currently performed directly: this implies uncertainty both in the value of the critical loads, and the precise locations of the critical cases in the load case space.

As described in chapter 8, this issue is exacerbated when including the effects of design variability, which implies the loads process is stochastic: as one is unable to fix the design, it behaves as a random variable, thus making the resulting loads for any given load case uncertain. We provide a rationale for assuming the marginal distributions of this stochastic loads process are normally distributed, and observe this to be a reasonable approximation in practice.

9.1.3 Objective 3: Development of Models for Loads-Structural Coupling Uncertainties

In chapter 7, we proposed a method for modelling uncertainty in the critical load cases. Using this model we are able to invoke a probability distribution over possible locations for the critical cases, which provides a powerful means of probabilistic load case downselection: given a sample of load cases, one can use the method to confidently rule out the presence of critical cases in certain regions of the search space, potentially saving many hours of simulation time

We also propose an explicit model for design variability induced uncertainty in the value of the loads in chapter 8, though further research is required to make the method useful in practice: since predictions of the marginal distribution parameters are imprecise, this results in overly diffuse predictive distributions, which compromises the utility of the method.

9.1.4 Objective 4: Robust Models of Uncertainty

In chapter 5 we outlined a method for constructing robust GP regression models which integrate over uncertainty in their hyperparameters, using considerations derived from theory in chapters 2 and 3. We hypothesised that this formulation of the models would yield reduced sensitivity to specifics of the data due to the use of

generalisable prior distributions, and more reliable quantification of predictive uncertainty by utilising a fully probabilistic treatment of the model hyperparameters. Chapter 6 shows this hypothesis to be true, and demonstrates effective performance of the models on a wide range of different synthetic and analytic engineering test functions.

9.2 Further Work

This work prompts a multitude of potential follow-up studies and questions.

Concerning the use of Bayesian Optimisation for critical case identification, a key limitation with the procedure demonstrated briefly in chapter 7 is that only a single IQ was considered at a time. A multi-objective Bayesian Optimisation framework would be ideal for approaching this problem. In particular, the multi-objective formulation of the PES [61] acquisition function, PESMO [67], is conceptually well suited to the problem, and can also be run in batches with some adjustments [68]. This relies on resolving the difficulties associated with implementing the original PES acquisition function, documented in appendix G.3.2.

Probabilistic representations of the *envelope*, rather than the critical cases themselves, could be considered by attempting to predict the value of the maximum of a GP model defined over the load case space rather than its location. Potential techniques to accomplish this are documented by Wang and Jegelka [64] and Ru et al. [65].

More research is required to understand the potential benefits of analysing the loads process as stochastic as described in chapter 8. Use of a latent covariate model such as that used by Wang and Neal [72] to model non-parametric non-stationary residuals may allow relaxation of the parametric assumption on the marginal distributions, and provide a way of correlating the marginals together.

Testing the models described in this document on larger and more complex load case spaces would be an interesting test of their capabilities. Extension of the optimisation search space to include more closely related *discrete* subspaces, such

as, for example, several different types of manoeuvre would be a highly relevant extension to the problem.

Finally, scaling the GP-regression derived methods in this document to larger, higher-dimensional datasets would significantly increase their industrial applicability. Wilson et al. [13] reviewed some of potential methods.

Appendices

Appendix A

Probabilistic Modelling Preliminaries

This appendix introduces Bayesian statistics as a language of uncertainty, along with several fundamental concepts required for a solid understanding of the models discussed later in this document.

A.1 Probability as a Language of Uncertainty

There are a multitude of frameworks for quantifying uncertainties, with perhaps the most widely known of these being the theory of probability.

Probabilities are classically interpreted as frequencies; the number of times in a long chain of events that a particular thing occurs relative to other things. This is intuitive, but presents conceptual difficulties when discussing the probability associated with events which are fundamentally not repeatable but clearly permit some kind of "degree of belief", such as the behaviour of the weather.

The uncertainty of repeatable events is sometimes referred to as *randomness* or *aleatory uncertainty*. Such uncertainties arise from processes which feature "natural" or – perhaps more specifically – *irreducible* sources of variation. Relevant engineering examples include small manufacturing variations or the velocity at a particular point in a turbulent flow, and in either case it is easy to imagine taking repeated measurements of the same quantities and obtaining slightly different results each time.

The uncertainty of deterministic events we have yet to observe, such as the weather tomorrow, is occasionally said to be due to *ignorance*, and is sometimes classified *epistemic*. The majority of uncertainties encountered in the analysis of computer codes epistemic in nature, for example the difference in performance between a simulated component and a realisation of the same component, or indeed the discrepancy between a simulator and a surrogate at a previously unvisited point.

By representing probability as a degree of belief rather than a frequency, Bayesian statistics is able to reconcile the apparent conceptual differences between aleatory and epistemic uncertainties. Conditional probability provides the mathematical machinery to make probabilistic statements about epistemic uncertainties given a current (and possibly limited) state of knowledge, along with the mathematical machinery to update and refine such statements as new information becomes available.

The ability to synthesise different types of information in a statistically consistent fashion in particular makes Bayesian statistics a powerful addition to the engineering

design toolbox, and permits past data and expert knowledge to be used together with data to construct more reliable models.

A complete, rigorous introduction to the theory of probability warrants its own textbook, and is outside the scope of this document. Readers are directed to Betancourt [73] for an excellent, self-contained theoretical introduction, and Gelman et al. [15, ch. 1] for an introduction in the context of applied modelling.

A.2 Probability Distributions and Random Variables

A probability distribution is a mathematical function which assigns a positive, scalar weighting to a set of "events", and integrates to unity over the space of all possible events. An event can be interpreted in context to be a possible "outcome" of an uncertain process. To "quantify an uncertainty" in the context of probabilistic modelling is hence to assign to it an appropriate probability distribution, thus implicitly designating the probability of all possible events.

Such events could include the toss of a coin, the roll of a die, the weather tomorrow, or whether a realised engineering design satisfies a certain performance target or not.

A.2.1 Probability of an Event

Let f be an arbitrary probability distribution, Ω be the space of all possible events, and ω be a single event. The probability, p, that an event, ω, occurs is denoted:

$$p(\omega) = f(\omega) \tag{A.1}$$

where $p(\omega)$ is a strictly positive scalar representing the probability (or "degree of belief") that the event represented by ω happens. Hence, f is a mapping between

the event space, Ω, and the positive numbers, $\mathbb{R}^+$:

$$f : \Omega \to \mathbb{R}^+ \tag{A.2}$$

We stress again that ω need not necessarily be the realisation of some inherently random process, and that it is perfectly acceptable to assign a probability to the outcome of a *deterministic* processes that we have yet to observe, such as the current temperature in a far away place.

A.2.2 Probability Mass Function

If Ω contains only discrete events, then f is a *probability mass function* (PMF), if:

$$\sum_{\omega \in \Omega} f(\omega) = 1 \tag{A.3}$$

A.2.3 Probability Density Function

If Ω is continuous, then f is a *probability density function* (PDF), if:

$$\int_{\Omega} f(\omega) d\omega = 1 \tag{A.4}$$

A.2.4 Random Variables

A random variable is a quantity which is permitted to take any value from a set of values. In the context of probability theory, a random variable represents possible realisations of an event, the relative "chance" of each of those events being determined by their probability. Random variables can be said to be "distributed according to" a (possibly unknown) probability distribution function, which designates the probability of all possible values the variable may take.

Let f be an arbitrary probability distribution and X be a random variable. To write

that X is distributed according to f, we write:

$$X \sim f \tag{A.5}$$

which is to imply that the probability of X being in some specific "state", ω, is given by the probability of that state as determined by f, i.e., $p(X = \omega) = f(\omega)$.

A.2.5 Random Vectors

A random variable is not restricted to be strictly univariate, and a single "event" may refer to a combination of two or more other events. Flipping two coins and observing that they both land heads up, for instance, may still refer to a single event.

A *random vector* is a random variable which is composed of two or more other univariate random variables. A d-dimensional random vector, $\mathbf{X}$, can be represented by:

$$\mathbf{X} \triangleq [X_1, X_2, \ldots, X_d] \tag{A.6}$$

where $X_1, X_2, \ldots, X_d$ are univariate random variables each possessing their own *marginal distribution*:

$$X_i \sim f_i \quad i = 1 \ldots d \tag{A.7}$$

where f_i is the marginal distribution for constituent variable i. In other words, if one were to draw many samples of $\mathbf{X}$, and then look *only* at element X_i, the distribution of those observations would be modelled by f_i.

A random vector, $\mathbf{X}$, is distributed according to a a *joint distribution* F:

$$\mathbf{X} \sim F_{X_1, X_2, \ldots, X_d} \tag{A.8}$$

which dictates how likely any particular combination of it's elements is. This is a function of the constituent marginal distributions *and* any interaction that occurs between them. Returning to the example of two coin flips, this could be considered a random vector composed of two constituent random variables representing the two individual coins.

A.3 Conditional Probability

Often, we are interested in the probability of one event *given* the occurrence of another. For example, suppose we have a 2 correlated random variables, X_1 and X_2, that can be modelled by the joint distribution $\mathbf{X} \sim F_{X_1 X_2}$. If we are able to observe X_2, we are often interested in analysing the *conditional* distribution for X_1, i.e. the marginal probability of X_1 *given* a specific value of X_2.

The probability of X given Y is denoted $p(X|Y)$. Betancourt [74] provides a short theoretical introduction to understanding some of the more nuanced properties conditional probability.

A.3.1 Bayes' Theorem

The joint probability of X and Y, $p(X, Y)$, is related to the conditional probability of X given Y, $p(X|Y)$, by multiplying the conditional probability by the marginal probability of observing Y, $p(Y)$:

$$p(X, Y) = p(X|Y)p(Y) \tag{A.9}$$

This applies identically in reverse i.e., to the probability of Y given X:

$$p(X, Y) = p(Y|X)p(X) \tag{A.10}$$

Combining equations (A.9) and (A.10) yields:

$$p(X|Y)p(Y) = p(Y|X)p(X) \tag{A.11}$$

which can be rearranged as desired to recover any conditional distribution given a joint distribution and the appropriate marginals. For example, given the probability of X given Y, $p(X|Y)$; the marginal distribution of X, $p(X)$; and the marginal distribution of Y, $p(Y)$, we are able to compute the probability of Y given X as follows:

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \tag{A.12}$$

Equation (A.12) is known as *Bayes' theorem*, and is core to performing *Bayesian inference*, which will be revisited shortly.

A.4 Parameters and Probabilistic Models

Probability distributions are often functions of a fixed number of variables that influence the behaviour of the function. These are referred to as *parameters*, and are usually denoted by a vector, θ.

If X is a random variable distributed according to f with parameters θ, we emphasise that the distribution, f, is a function of θ:

$$X \sim f(\theta) \tag{A.13}$$

in which case the probability distribution, f, is said to be *parametric*. Non-parametric distributions, by contrast, are functions of a variable (possibly infinite) number of parameters.

The probability of observing X given it is distributed by a parametric model with parameters θ is expressed as $p(X|\theta)$, i.e. the probability of X is *conditional* on θ.

This notation is observed to be consistent with θ itself being a random variable, and is in fact one of the central tenets of Bayesian statistics: while the parameters of a model may not be random in the sense that they are realisations of some uncertain process, they can (and should) be *modelled* as random variables given we are unable to observe them directly. This includes assignment of their own (possibly unknown) probability distribution.

A.4.1 Types of Parametric Distribution

Parametric distributions are mathematically convenient as they permit a neat algebraic description of an uncertainty (that is, they can be analysed without sampling). Such distributions also tend to originate from models of a particular phenomenon (for example, the number of events occurring with fixed rate in a particular time is modelled by a *Poisson* distribution), giving them an intuitive (physical) interpretation, and often providing a rationale for their usage in certain contexts.

The Stan language functions reference [75] provides a comprehensive reference to many discrete (sections 10-14) and continuous (sections 15-25) parametric distributions.

A.4.2 Likelihood Functions

Let X be a random variable distributed according to some probability distribution function f with parameters θ. The probability of *observing* $X = \omega$ is given by:

$$p(X = \omega|\theta) = f(\omega|\theta)p(\theta) \tag{A.14}$$

where $p(\theta)$ is the probability of the parameters.

In many cases, observation of a random variable, X, represents a sample from a population (a "piece of data"). With reference to equation (A.14), the probability of observing $X = \omega$ (given we *assume* a probability distribution of form f) is a function of ω, conditional on θ.

It is also useful to think of the probability of some data, ω, as a function of θ. This is termed the *likelihood* of θ, denoted as follows:

$$\mathcal{L}(\theta|\omega) = p(X = \omega|\theta) = f(\omega|\theta) \tag{A.15}$$

It is important to distinguish the *likelihood* from *probability*. While the two are sometimes synonymous in non-specialist literature, the *likelihood of a parameter* refers specifically to the probability of the *data* given the parameter, while *probability of a parameter* refers to the probability of that parameter given the data.

A likelihood function (i.e. a means of expressing the probability of some observed data) is essential to constructing a probabilistic model, and the choice of likelihood function contributes major modelling assumptions upon which subsequent analysis is conditional.

A.5 Inference

Given a parametric model for the random behaviour of a system, we often wish to fit it to a particular set of data. This implies *inference* of the parameters governing such behaviour, using available observations as evidence. For example, supposing one is able to appeal to the central limit theorem to assert that a quantity is normally distributed, the parameters of this distribution must still be identified for it to be useful.

Let $\mathbf{x}$ represent n samples of some random variable, X, which is assumed to be distributed according to some probability distribution function f, with parameters θ, i.e. $X \sim f(\theta)$. Since θ cannot be observed directly, it must be inferred, using $\mathbf{x}$ as evidence. This is equivalent to identifying the distribution of $p(\theta|\mathbf{x})$.

Recalling Bayes' theorem, described by equation (A.12), this conditional probability

can be expressed as:

$$p(\theta|\mathbf{x}) = \frac{p(\theta|\mathbf{x})p(\theta)}{p(\mathbf{x})} \tag{A.16}$$

which can be broken down as follows:

- $p(\mathbf{x}|\theta)$ - the **marginal likelihood** or **evidence** denotes the probability of the data given the parameters. With reference to section (A.4.2), this is obtained by evaluating the likelihood function for the chosen model given specific values of the parameters, and is equivalently denoted $\mathcal{L}(\theta|\mathbf{x})$.
- $p(\theta)$ - the **prior** expresses the probability of the parameters in the absence of any other information. In other words, the prior encapsulates any beliefs about the value of θ before any data is considered.
- $p(\mathbf{x})$ - the **probability of the data** expresses the probability of observing the data which used as evidence for θ.
- $p(\theta|\mathbf{x})$ - the **posterior** expresses the probability of the parameters, given the data, for that particular likelihood function. This is what we aim to identify.

In many applications, the probability of the data is simply a normalisation constant (ensuring $p(\theta|\mathbf{x})p(\theta)$ integrates to 1), since the observations can be considered deterministic. Equation (A.16) is often written as:

$$p(\theta|\mathbf{x}) \propto p(\theta|\mathbf{x})p(\theta) \tag{A.17}$$

which is to say that $p(\theta|\mathbf{x})$ is equivalent to $p(\theta|\mathbf{x})p(\theta)$ up to a multiplicative constant.

Identification of the posterior distribution for θ given $\mathbf{x}$ consequently requires computation of the marginal likelihood and the prior.

The most important point to note at this stage is that θ can only ever be *approximated* as a consequence of inferring it from a finite amount of data.

A.5.1 Computation

In general, the posterior distribution does not permit an analytical representation, except in the case of specific *conjugate* prior and likelihood functions. Such particularities can be limiting for many practical problems, as they tend to force the use of specific combinations of likelihood and/or prior distributions for algebraic convenience, rather than encoding problem specific information.

A less elegant but considerably more powerful approach is to approximate the posterior using sampling, drawing samples from a distribution proportional to $p(\theta|\mathbf{x})p(\theta)$ using Markov-Chain Monte-Carlo (MCMC). Gelman et al. [15, ch. 10-11] provide detailed and accessible exposition on the use of MCMC to accomplish this. Such methods allow more arbitrary choices of likelihood functions and/or prior distributions, permitting a practitioner to tailor these important assumptions to the specifics of the problem without worrying about satisfying conjugacy requirements.

Appendix B

Basics of Gaussian Process Regression

This appendix provides a verbose introduction to Gaussian Process regression as a technique for surrogate modelling. We provide a high-level, intuitive description of the method and describe the basic computational steps required to perform GP regression in practice.

B.1 Introduction

For the purposes of example, let f be some scalar-output function we are interested in constructing a model for. The function maps a D-dimensional vector of real numbers, $\mathbf{x}$, onto a single, real number, y:

$$f : \mathbb{R}^D \to \mathbb{R} \tag{B.1}$$

The "support" of a GP – that is, the range of permissible inputs to the function one wishes to model – is referred to as the *index*. In the majority of cases this is a subset of the real numbers. For the case of a function defined over a D-dimensional index, as above, the index is denoted $\mathcal{X}$ (sometimes emphasising that it is a subset of the $\mathbf{R}^D$ via explicitly stating $\mathcal{X} \subseteq \mathbb{R}^D$, though this is often superfluous).

B.1.1 Distributions Over Random Functions

In practical terms, a GP is a probability distribution, and the functions over which the GP invokes a distribution are controlled by its parameters. All GPs have two parameters: the mean function, $m(\mathbf{x})$, and the covariance kernel, $k(\mathbf{x}, \mathbf{x}')$. These will be introduced in section B.2.

To describe a random function, $\mathbf{f}$, observed at some points $\mathbf{X}$, as "distributed as" a GP, one writes:

$$p(\mathbf{f}|\mathbf{X}) \sim \mathcal{GP}\left(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X})\right) \tag{B.2}$$

which (literally) states that the probability of observing $\mathbf{f}$ at the points contained in $\mathbf{X}$ is "distributed as" a GP.

A random function, $\mathbf{f}$, behaves like a lazily-evaluated, "infinitely long" vector, leading GP distributions to be sometimes referred to as infinite-dimensional analogues of the Multivariate Normal (MVN) distribution: while there is a computational limit on how *many* points, $\mathbf{x}$, at which we can "observe" the GP distribution at any

given time, there are no limits on *which* points we could choose to observe from the index.

A single sample from a GP (alternatively, a "realisation" or "sample path") is thus a Gaussian *random vector*, with mean vector, $\mathbf{m}$, and covariance matrix, $\mathbf{K}$, which – by inspecting equation (B.2) – are noted to be functions of the points in the index at which that particular sample is observed.

For example, if we observe a random function, $\mathbf{f}$, at N points, $X = [\mathbf{x}_1 \dots \mathbf{x}_N]$, then the observed distribution of $\mathbf{f}$ (at these points) will have an MVN distribution:

$$p(\mathbf{f}|X) \sim \text{MVN}(\mathbf{m}, \mathbf{K}) \tag{B.3}$$

A single element of the random vector, $\mathbf{f}$, is a univariate Gaussian random variable, referred to as a *marginal distribution* (or just "marginal"). The marginals of a random vector refer to the distributions of the vector's constituent elements when considered in isolation. For a GP, the marginals of an arbitrary random function are univariate Gaussian, with mean and standard deviation corresponding to the respective element of $\mathbf{m}$ and $\text{diag}(\mathbf{K})$. For instance, the i^{th} element of $\mathbf{f}$, denoted f_i, describes the marginal distribution at $\mathbf{x}_i$, and has the following distribution:

$$f_i \sim \mathcal{N}(\mathbf{m}_i, \mathbf{k}_i)$$

where $\mathbf{m}_i$ represents the i^{th} value of $\mathbf{m}$ and k_i represents the i^{th} element of $\mathbf{K}$'s leading diagonal.

A visualisation of a simple GP distribution and an arbitrary marginal for illustrative purposes is shown in figure (B.1).

B.2 Parameters

With reference to equation (B.2), a GP has two parameters: $m(\mathbf{x})$, the *mean function*; and $k(\mathbf{x}, \mathbf{x}')$, the *covariance kernel*. Both the mean function and covariance

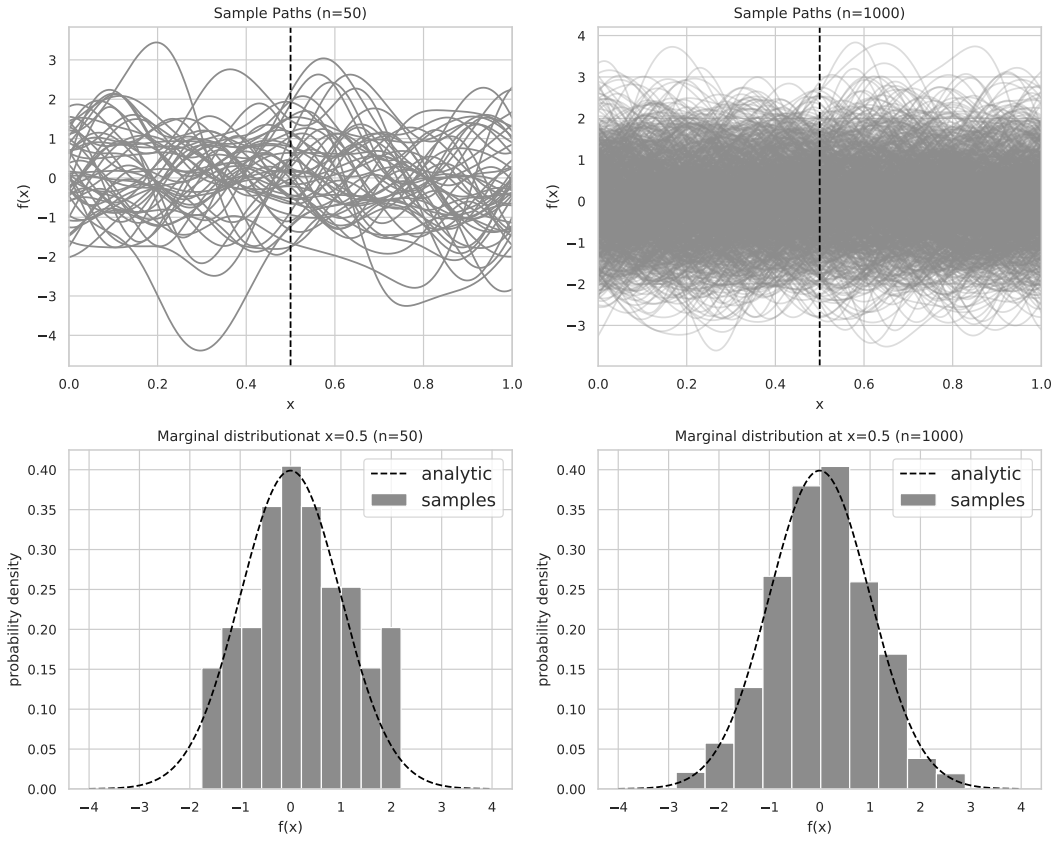

Figure B.1: Visualisation of GP sample paths (top row) and marginal distributions corresponding to $x = 0.5$ (bottom row). The 1D marginal distributions can be interpreted as the distribution at a "slice" corresponding to a particular value of x, indicated by the dashed line. The analytical marginals, which correspond to an infinite sample size, are also shown, and are obtained by evaluating the mean function and covariance kernel of the GP at $x = 0.5$.

kernel are functions of the index.

B.2.1 Mean Function

The mean function of a GP maps a point in the index to a scalar value:

$$m : \mathbb{R}^D \to \mathbb{R} \tag{B.4}$$

$m(\mathbf{x})$ describes the mean of the GP distribution at every point in the index: this can be thought of as encoding the "best guess" (or more precisely, *expectation*) of the value of the function one is modelling as a GP in the absence of any data. The mean function is occasionally designated as "uninteresting", as it can be subtracted from the data prior to performing any analysis without loss of generality. Equivalently, this implies GPs with parametrised mean functions are in fact zero-mean models of the residuals obtained by subtracting the mean function from the data:

$$p(f(\mathbf{X}) - m(\mathbf{X})|\mathbf{X}) \sim \mathcal{GP}(\mathbf{0}, k(\mathbf{X}, \mathbf{X}')) \tag{B.5}$$

The mean function is still useful. For example, if one knows that the function under consideration features perturbations about some unknown linear trend, a linear mean function might be used to capture this prior information, with the regression coefficients of the mean function included in the GP model. It is noted that arbitrarily complex mean functions are theoretically possible.

Table B.1 includes several of the most common mean functions used for surrogate modelling applications.

In general, a more simple mean function implies more properties of the data generating mechanism are left for the GP to "discover". This can be important if only "abstract" properties of the target function can be reasoned with, in which case one might want to avoid imposing any parametric assumptions on functions one considers for the data by using a zero mean function.

$m(\mathbf{x}) =$	*Type*	*Notes*
0	Zero	
c	Constant	
$\beta\hat{\mathbf{x}}$	Linear	$\hat{\mathbf{x}} := [1, x_0, \dots x_D]$

Table B.1: Common mean functions for GPs used in surrogate modelling applications. For the linear and polynomial mean functions, $\hat{\mathbf{x}}$ is a vector of regression variables, and β a vector of regression coefficients. $\hat{\mathbf{x}}$ can be tailored to include only specific input dimensions if desired.

B.2.2 Covariance Kernel

The covariance kernel of a GP maps a pair of points from the index to a scalar value (which is usually but not necessarily positive):

$$k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R} \tag{B.6}$$

The covariance kernel (often just "kernel"), $k(\mathbf{x}, \mathbf{x}')$, is responsible for designating the covariance between any two points in the index. Intuitively, this can be thought of as a function which describes the similarity between any two possible combinations of points: more "similar" points can then be interpreted to vary together (i.e., to exhibit greater correlation), such that some "structure" is imposed on functions defined by the GP via the kernel. The kernel consequently controls which "types" of functions a GP encodes a distribution over.

In order for an arbitrary function, g, to be classed as a covariance kernel, the distance matrix obtained by applying g pairwise between a set of points must be a valid *covariance matrix*: that is, the resulting matrix must be symmetric in it's leading diagonal and positive-definite, implying it is invertible.

B.3 Gaussian Process Regression

GPs are used for regression by conditioning a GP (prior) distribution on some observed data, utilising the resulting posterior distribution for prediction. This conditioning can be performed analytically due to assuming the distribution of functions capable of representing the data is jointly Gaussian.

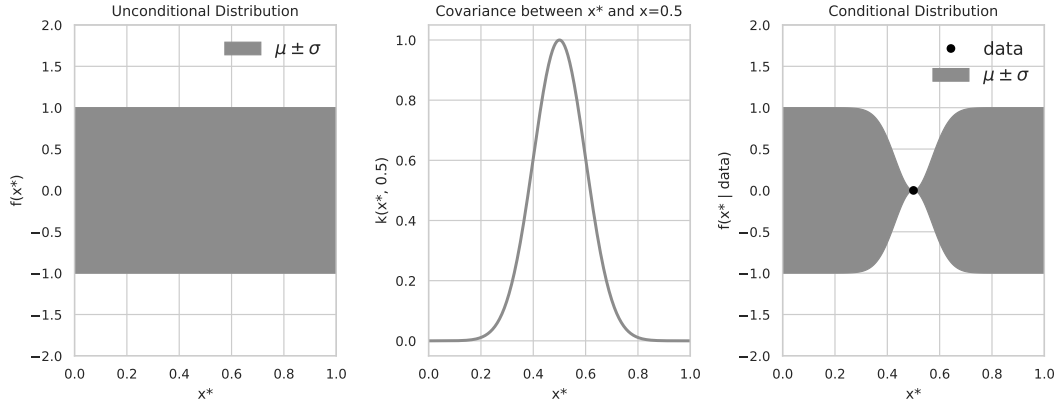

Figure B.2: Visualisation of the process of conditioning an unconditional GP distribution (left), given the cross-covariance with some observed data (middle), to produce a conditional (or "predictive") distribution (right).

Let a dataset be represented by $\mathcal{D}$. A dataset is assumed to consist of N tuples, each consisting of a D-dimensional vector, $\mathbf{x}$, representing an input (or sample point); and a scalar value, y, representing the corresponding output of the function of interest at $\mathbf{x}$, i.e., $y_i = f(\mathbf{x}_i)$.

Vertically stacked arrays of $\mathbf{x}$ and y are represented by $\mathbf{X} \triangleq [\mathbf{x}_1 \dots \mathbf{x}_N]^\intercal$ and $\mathbf{y} \triangleq [y_1 \dots y_N]^\intercal$, respectively, where X has size $N \times D$ and $\mathbf{y}$ has size N.

Finally, let $\mathbf{X}^*$ be a vertically stacked array of M *new* D-dimensional points from the index, $\mathbf{X}^* \triangleq [\mathbf{x}_1^* \dots \mathbf{x}_M^*]^\intercal$, at which we are interested in making a prediction conditional on (or "given") the observed data, $\mathcal{D}$.

A GP is used for prediction by first constructing an unconditional GP defined at the points one is interested in making a prediction at, $\mathbf{X}^*$, then conditioning this distribution on the observed data using the unconditional GP defined at $\mathbf{X}$. The process of conditioning on the observed data is sometimes described as "explaining away" a portion of the unconditional variance, the remaining "unexplained" variance constituting the predictive distribution. A pedagogic visualisation for a 1D function is shown in figure (B.2).

In the following sections we introduce arbitrary arrays of points, A and B, to more clearly distinguish between which sets of points different covariances are to be computed.

B.3.1 Unconditional Gaussian Process

An unconditional GP distribution is obtained by evaluating the mean function, $m(\cdot)$, and covariance matrix, $k(\cdot,\cdot)$, for each of the points under consideration.

To compute the parameters for an unconditional GP over an arbitrary $N \times D$ array of points $\mathbf{A}$, we evaluate the mean function $m(\cdot)$ for each of the N points in $\mathbf{A}$ to produce a length N mean vector $\mathbf{m}_A$:

$$\mathbf{m}_{A_i} = m(\mathbf{a}), \quad i = 1 \dots N \tag{B.7}$$

and covariance kernel, $k(\cdot,\cdot)$, pairwise for each of the points in $\mathbf{A}$ to produce an $N \times N$ covariance matrix, $\mathbf{K}_{AA}$:

$$\mathbf{K}_{AA,(ij)} = k(\mathbf{a}_i, \mathbf{a}_j), \quad i = 1 \dots N, \quad j = 1 \dots N \tag{B.8}$$

where $\mathbf{a}$ is a single row of $\mathbf{A}$ representing a D-dimensional point in the index.

B.3.2 Prediction

Predictions are made using a GP by conditioning the distribution over some points of interest on observed data. This implies a prediction is a random variable. Let $\mathbf{X}^*$ be an array of M points at which we are interested in making a prediction. The predictive distribution of a GP at $\mathbf{X}^*$, conditional on data, $\mathcal{D}$, is:

$$p(\mathbf{y}^*|\mathbf{X}^*, \mathcal{D}) \sim \mathcal{GP}(\mathbf{m}^*, \mathbf{K}^*) \tag{B.9}$$

where $\mathbf{y}^*$ is a length M random vector representing a prediction, $\mathbf{m}^*$ is the conditional (or *predictive*) mean vector, and $\mathbf{K}^*$ is the conditional covariance matrix.

Since a GP evaluated over a finite subset of points is a multivariate normal distribution, closed form expressions can be derived (see [76, p.116-117]) for both the conditional mean and conditional covariance matrix, such that the predictive distri-

bution described by equation (D.13) can be obtained analytically.

Let $\mathbf{K}_{AB}$ represent the $M \times N$ *cross-covariance matrix* between two arrays of points, $\mathbf{A}$ and $\mathbf{B}$, obtained by evaluating k pairwise between the rows of the $M \times D$ array $\mathbf{A}$ and $N \times D$ array $\mathbf{B}$:

$$\mathbf{K}_{AB,(ij)} = k(\mathbf{a}_i, \mathbf{b}_j), \quad i = 1 \dots M, \quad j = 1 \dots N \tag{B.10}$$

Similarly, let $\mathbf{m}_{(\cdot)}$ be the mean vector obtained by evaluating $m(\cdot)$ for each element of the subscripted variable, and $K_{(\cdot,\cdot)}$ the covariance matrix obtained by evaluating $k(\cdot, \cdot)$ pairwise between each element of the subscripted variables (noted to be square and symmetrical if the two arguments are identical).

Representing the inverse of the data covariance matrix by $\Omega \triangleq [\mathbf{K}_{\mathbf{XX}}]^{-1}$, the predictive mean vector, $\mathbf{m}^*$ is given by:

$$\mathbf{m}^* = \mathbf{m}_{\mathbf{X}^*} + K_{\mathbf{X}^*,\mathbf{X}} \cdot \Omega \cdot (\mathbf{y} - \mathbf{m}_{\mathbf{X}}) \tag{B.11}$$

and predictive covariance matrix, $\mathbf{K}^*$, by:

$$\mathbf{K}^* = K_{\mathbf{X}^*\mathbf{X}^*} - \mathbf{K}_{\mathbf{X}^*,\mathbf{X}} \cdot \Omega \cdot K_{\mathbf{X},\mathbf{X}^*} \tag{B.12}$$

where it is noted that $K_{\mathbf{X}^*,\mathbf{X}} = K_{\mathbf{X},\mathbf{X}^*}^{\intercal}$.

B.3.3 Computation In Practice

It is often beneficial to avoid explicit matrix inversions where possible. Even though every valid covariance kernel, k, ensures that the resulting matrix obtained by evaluating k pairwise between a set of points is an invertible matrix, numerical instabilities are an issue when performing such computations in practice.

A more stable approach to computing equations (B.11) and (B.12) is to work with the Cholesky factor of the data covariance matrix. The lower Cholesky factor is

defined for an arbitrary, symmetric, positive-definite, real matrix, $\mathbf{A}$, as:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{L}^{\intercal} \tag{B.13}$$

from which the dot product of the inverse of $\mathbf{A}$ with an appropriately sized matrix $\mathbf{B}$ can be obtained by solving the following linear system for $\mathbf{C}$:

$$\mathbf{L} \cdot \mathbf{C} = \mathbf{B} \tag{B.14}$$

which we denote $\mathbf{L} \backslash \mathbf{B}$.

It follows that given the lower Cholesky factor $\mathbf{K_{XX}} = \mathbf{L_{XX}} \cdot \mathbf{L_{XX}^{\intercal}}$, we may reformulate equations (B.11) and (B.12) as:

$$\mathbf{m}^* = \mathbf{m}_{\mathbf{X}^*} + \mathbf{K}_{\mathbf{X}^*,\mathbf{X}} \cdot (\mathbf{L_{XX}} \backslash (\mathbf{y} - \mathbf{m_X})) \tag{B.15}$$

and

$$\mathbf{K}^* = \mathbf{K}_{\mathbf{X}^*,\mathbf{X}^*} - \mathbf{K}_{\mathbf{X}^*\mathbf{X}} \cdot (\mathbf{L_{XX}} \backslash K_{\mathbf{X}\mathbf{X}^*}) \tag{B.16}$$

respectively.

Appendix C

Gaussian Process Regression Extensions

This appendix contains a number of non-standard adaptations to the standard GP regression model which may be useful for specific applications.

C.1 Composite Kernels

Under certain circumstances, kernels can be combined to produce "composite" kernels. Such kernels are capable of encoding distributions over functions which are sums and/or products of constituent "base" functions, optionally only including certain orders of interaction. A "standard" kernel, for comparison, is equivalent to a product kernel featuring all orders of interaction.

Encoding such structure a-priori (in contrast to attempting infer the properties of the composite function) is often beneficial from a modelling perspective. If the function of interest is known to be composed as sums or products of constituent functions, including this structure explicitly in the kernel is advantageous. Even if such an assumption cannot be made confidently, many practical benchmark problems can be observed to exhibit some additive structure, provided a flexible enough method can be deployed to infer it [77].

A detailed discussion of composite kernels can be found in [14, ch. 2]. *Additive Gaussian Processes* (AGPs; see [77] and [14, ch. 6]) are a special type of composite kernel, designed to infer and exploit additive structure automatically. This is particularly beneficial for higher dimensional problems as "long distance" covariances can be more easily preserved: for product kernels, points must be "similar" (as designated by the kernel) in all input dimensions to receive high covariance, while for additive kernels, points may exhibit relatively high covariance if only some input dimensions are designated as similar. A visual example is provided in figure (C.1).

C.2 Encoding Noise

Sometimes, we wish to model functions which feature noise. This arises when the function outputs that we observe are imprecise for some reason: an observed quantity might be corrupted, or there may be an unknown residual error due to some missing piece of information. The latter is more common in the context of surrogate modelling, as physics-based computer models are usually considered deterministic.

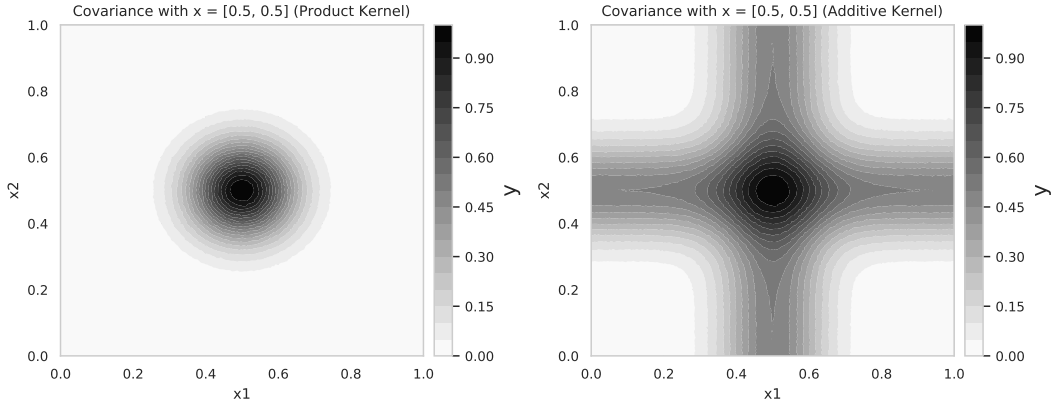

Figure C.1: Comparison of the covariance structures designated by product (left) and additive (right) covariance kernels for a 2D index. Both plots show the covariance between a fixed point, $\mathbf{x}^* = [0.5, 0.5]$, and another (arbitrary) point in the index, $\mathbf{x} = [x_1, x_2]$. The kernels have a marginal standard deviation of 1 and an isotropic lengthscale of 0.1. The product kernel (left) is noted to designate high covariance only if *both* x_1 and x_2 are close. The additive kernel (right), by contrast, is able to designate higher covariance if *either* x_1 or x_2 are similar.

Gaussian (white) noise can be included on any "base" kernel by additively superimposing a kernel which models the desired noise. For example, to model homoscedastic white noise of standard deviation σ, we can use the stationary *white noise kernel*:

$$k_{\text{wn}}(\mathbf{x}, \mathbf{x}'|\theta) = \begin{cases} \sigma^2 & \text{if } \mathbf{x} \equiv \mathbf{x}' \\ 0 & \text{otherwise} \end{cases} \tag{C.1}$$

k_{wn} produces diagonal covariance matrices with σ^2 along the leading diagonal, and has a single hyperparameter $\theta \triangleq [\sigma]$.

To formulate a "noisy" implementation of an arbitrary base kernel, k, we combine k and k_{wn} additively to produce a noisy version of k, denoted k_{noisy}:

$$k_{\text{noisy}}(\mathbf{x}, \mathbf{x}'|\theta) = k(\mathbf{x}, \mathbf{x}'|\theta_1) + k_{\text{wn}}(\mathbf{x}, \mathbf{x}'|\theta_2) \tag{C.2}$$

where the hyperparameters of k_{noisy} obtained by concatenating the hyperparameters of the base kernel and the hyperparameters of the noise kernel, i.e., $\theta \triangleq [\theta_1, \theta_2]$.

The noise modelled by the kernel described by equation (C.1) is *Gaussian* as we

simply add extra variance of magnitude σ^2 to the GP marginals (that is, the leading diagonal of the covariance matrix), which are Gaussian by definition. Non-Gaussian noise requires transformation of the marginals (such that the resulting stochastic process is no longer a *Gaussian* process) and is discussed in section C.4.

C.3 Non-Stationary GP Regression

For a *non*-stationary GP, the abstract properties of the functions over which the model invokes a distribution are not necessarily constant over the index. In practice, this tends to mean that the hyperparameters of such a kernel are variable over the index.

To perform non-stationary GP regression, one consequently requires a model for the variation of these hyperparameters across the index. This model may even be a GP itself, and inference of this model for variation implies a significantly more challenging problem than for stationary GP regression: models of such variation require their own parameters, which must be inferred, in effect multiplying the total number of (hyper-)parameters the top model must consider. This can have important consequences on the ability to uniquely identify suitable hyperparameters.

The most important non-stationary effects tend to be those induced by kernel lengthscales and input-dependent noise, as these have the most significant effects on the predictive capabilities of the model; the latter due to influence on the ability to identify the correct model hyperparameters. An effective method for modelling lengthscale non-stationarity is using input warping [78], in which the model inputs are transformed by a nonlinear warping function before interacting with the kernel. This method permits a probabilistic implementation, such that different types of non-stationarity can be integrated over; and is intuitive, straightforward to implement, and permits an implicit prior distribution over different types of warping, should such information be available.

Input dependent (*hetereoscedastic*) noise can be modelled using a second GP determining the standard deviation of (Gaussian) noise as a function of the index [79],

or by using a latent-variable formulation such as that described by Wang and Neal [72], with the latter noted to permit non-Gaussian noise.

The inclusion of non-stationary effects has been shown to have benefits in terms of performance. Toal and Keane [80] showed inclusion of non-stationary effects provided enhanced performance on certain surrogate-based optimisation problems relative to stationary models, though noted that their results were problem dependent, perhaps due to difficulty identifying the correct non-stationary behaviour for complex functions. The input warping strategy of Snoek et al. [78] was shown to provide *consistently* superior performance relative to stationary models for several Bayesian Optimisation problems, due to being able to navigate any identifiability problems by integrating the predictive distribution over many types of potential non-stationary effects.

C.4 Non-Gaussian Marginals

To produce *non-Gaussian* marginals, required when modelling non-Gaussian noise, we must transform the GP marginals to the desired distribution, noting critically that the transformed distribution will no longer be a Gaussian process. A simple way of accomplishing this for (transformed) distributions which permit a probability distribution function is to transform the GP marginals onto $(0, 1)$ using the Gaussian CDF, and then to transform this uniform distribution into the desired marginals using the appropriate inverse CDF.

For the purposes of example, suppose we have a normally distributed random variable, Y, with mean μ and standard deviation σ:

$$Y \sim \mathcal{N}(\mu, \sigma)$$

we transform Y into a uniform random variable, U:

$$U = \Phi(Y|\mu, \sigma)$$

given $\Phi(\cdot|\mu, \sigma)$ is the Gaussian CDF parametrised by μ and σ, respectively. Finally, we transform U into a random variable, Z, with the desired parametric distribution:

$$Z = F^{-1}(U|\omega)$$

where F^{-1} is the inverse CDF of a probability distribution, F, with parameters ω. It is noted that identification of ω is typically challenging.

C.5 Gradient Enhanced GP Regression

Since differentiation in a linear operator, the derivative of a GP is another GP (see, for example, [81]). This can be utilised, in that derivative observations can provide extra information about the target function at observed sample points. Conditioning on this extra information when constructing the predictive distribution of the model provides a reduction in posterior variance, thus conferring a more precise predictive distribution relative to a model in which this information is not present.

Gradient enhanced GP regression was explored by Solak et al. [19] in the context of modelling dynamic systems and by Dwight and Han [82] in surrogate modelling of computational fluid dynamics simulators featuring adjoint gradients.

Appendix D

Supplementary Material For Loads Structural-Coupling

This appendix contains detailed documentation for technical procedures covered only briefly in chapter 4.

D.1 Principal Component Analysis via Singular Value Decomposition

We perform PCA using singular value decomposition (SVD). The principal components of the $M \times N$ data matrix, $\mathbf{Y}$, are the eigenvectors of its sample covariance matrix, denoted $\mathbf{C}$:

$$\mathbf{C} = \mathbf{Y}^\intercal \mathbf{Y}/(N-1) \tag{D.1}$$

Since $\mathbf{C}$ is (by definition) a covariance matrix, it is symmetrical and positive semi-definite, and thus permits diagonalisation:

$$\mathbf{C} = \mathbf{V} \cdot \mathrm{diag}(\lambda) \cdot \mathbf{V}^\intercal \tag{D.2}$$

where $\mathbf{V}$ is an $M \times M$ matrix of columnwise eigenvectors, and $\mathrm{diag}(\lambda)$ is an $M \times M$ diagonal matrix of eigenvalues. The SVD of $\mathbf{Y}$ is:

$$\mathbf{Y} = \mathbf{U}\mathbf{S}\mathbf{V}^\intercal \tag{D.3}$$

where $\mathbf{S}$ is a diagonal matrix of singular values, and $\mathbf{U}$ is a unitary matrix (i.e. $\mathbf{U}\mathbf{U}^\intercal = \mathbf{U}^\intercal\mathbf{U} = \mathbf{I}$). Combining equations (D.3) and (D.1) gives:

$$\mathbf{C} = \mathbf{V}\mathbf{S}\mathbf{U}^\intercal \cdot \mathbf{U}\mathbf{S}\mathbf{V}^\intercal/(N-1) \tag{D.4}$$

which with reference to equation (D.2) implies:

$$\mathbf{C} = \mathbf{V}\frac{\mathbf{S}^2}{N-1}\mathbf{V}^\intercal \tag{D.5}$$

yielding the following relationship between the singular values, $\mathbf{S}$, and eigenvalues

of the sample covariance matrix, λ:

$$\lambda = \text{diag}(\mathbf{S})/(N-1) \tag{D.6}$$

One performs dimension reduction to R dimensions using PCA by retaining the eigenvectors corresponding to only the first R eigenvalues (in descending order). Stacking the eigenvectors corresponding to the remaining eigenvectors into an $M \times R$ array yields a matrix, $\mathbf{P}$. One can then transform an arbitrary set of points in the original space, expressed as an $N_{\text{new}} \times M$ matrix, $\mathbf{Y}^*$, onto the PCA basis, by taking their dot product with $\mathbf{P}$:

$$\mathbf{Q}^* = \mathbf{Y}^*\mathbf{P} \tag{D.7}$$

where $\mathbf{Q}^*$ is a representation of the points $\mathbf{Y}^*$ in the PCA basis described by $\mathbf{P}$. It follows that an *approximation* to $\mathbf{Y}^*$ can be recovered from $\mathbf{Q}^*$ by taking the dot product with the transpose of $\mathbf{P}$, i.e., $\mathbf{Y}^* \approx \mathbf{Q}^*\mathbf{P}^\intercal$.

D.2 Scaling

PCA is sensitive to the scaling of the data. The method can be interpreted to "identify the directions of maximum variation" in the data, and is consequently biased towards recovering variation in high-variance columns of the data matrix if no scaling is applied.

To help remedy this behaviour, it is common to apply a scaling function to the columns of the data matrix first, such that all columns have approximately zero mean and unit standard deviation. Let $\mathbf{y}_m$ be the m^{th} column of $\mathbf{Y}$. The canonical scaling technique is to subtract the column sample mean and divide by the column sample standard deviation. However, the sample standard deviation is not a *robust* statistic, and can thus be sensitive to outliers. We opt to use the Maximum Absolute

Deviation (MAD) instead. Our scaling function is thus:

$$\phi(\mathbf{y}_m) = \frac{\mathbf{y}_m - \bar{\mathbf{y}}_m}{\text{MAD}(\mathbf{y}_m)} \tag{D.8}$$

where $\bar{\mathbf{y}}_m$ is the sample mean of $\mathbf{y}_m$ and $\text{MAD}(\mathbf{y}_m)$ is the median of the absolute deviations from the sample mean:

$$\text{MAD}(\mathbf{y}_m) = \text{Md}(|\mathbf{y}_m - \bar{\mathbf{y}}_m|) \tag{D.9}$$

given $\text{Md}(\cdot)$ represents the median of its argument.

It is worth noting that when "decompressing" values from the PCA basis space back into the original data space, one must remember to properly rescale the resulting transformed values using the same values that were used to compute the PCA basis. A set of N_{new} arbitrary points in the transformed basis of dimension R, represented by an $N_{\text{new}} \times R$ array, $\mathbf{Q}^*$, is transformed back into an $N_{\text{new}} \times M$ array of points in the original data space, $\mathbf{Y}^*_{\text{new}}$, by taking the dot product with the transpose of the PCA rotation matrix and applying the reverse of the desired scaling function (i.e., multiplication of each column by the MAD and addition of the sample mean):

$$\mathbf{Y}^*_{\text{new}} = \phi^{-1}(\mathbf{Q}^*\mathbf{P}^\intercal) \tag{D.10}$$

where $\phi^{-1}(\cdot)$ represents application of the appropriate rescaling functions to the columns of the argument. Assuming $\mathbf{W} = \phi(\mathbf{Y})$, one recovers $\mathbf{Y}$ by:

$$\mathbf{Y} = \phi(\mathbf{W})^{-1} = \mathbf{Y} * \text{MAD}(\mathbf{Y}) + \overline{\mathbf{Y}} \tag{D.11}$$

D.3 Selecting the Number of Basis Terms

A common strategy to choosing R is to propose a cut-off point for the desired proportion of total variation in the data one seeks to explain using the compressed basis.

This fraction is equivalent to the ratio between the sum of the first R eigenvalues and the total sum of all of the eigenvalues of the data covariance matrix. This approach, however, does not take into account subsequent *rescaling* of the data, and is thus not necessarily guaranteed to distribute the residual variance in any particularly predictable manner after rescaling.

Providing access to a test set, a more robust choice is to analyse the reconstructed data for different values of R. In this case it is imperative that $\mathbf{P}_{\text{test}}$ is constructed using only the training data (including columnwise normalisation), and that the test data is scaled and rescaled by the same functions as the training data.

Let $\mathbf{Y}_{\text{test}}$ be an $N_{\texttt{test}} \times M$ array of test points. To obtain the residuals associated with compressing $\mathbf{Y}_{\text{test}}$ onto R basis terms, we first require the data as re-constructed from the new basis (i.e., compression and subsequent recompression):

$$\mathbf{Y}^*_{\text{test}} = \phi^{-1}\left(\left(\phi(\mathbf{Y}_{\text{test}}) \cdot \mathbf{P}_{[:R]}\right) \cdot \mathbf{P}^{\intercal}_{[:R]}\right) \tag{D.12}$$

where $\mathbf{P}_{[:R]}$ implies the $R \times M$ matrix obtained by retaining only the first R rows of the PCA matrix, $\mathbf{P}$; $\phi(\cdot)$ represents a scaling operation; and $\phi^{-1}(\cdot)$ a rescaling operation (see section D.2).

$\mathbf{Y}^*_{\text{test}}$ is computed for each load case, and the reconstructed envelope, $\overline{\mathbf{Y}}^*_{\text{test}}$, is calculated via equation (4.11).

A matrix of residuals, $\mathbf{R}$, is obtained by comparing the reconstructed envelope to the actual envelope: $\mathbf{R} = \overline{\mathbf{Y}}_{\text{test}} - \overline{\mathbf{Y}}^*_{\text{test}}$.

Mean residuals for the normalised envelope as a function of R are plotted in figure D.1. The torque measurements can be seen to exhibit the greatest normalised mean residual error (note the scales of the y-axes). Increasing the number of basis terms reduces the mean residual error, as expected.

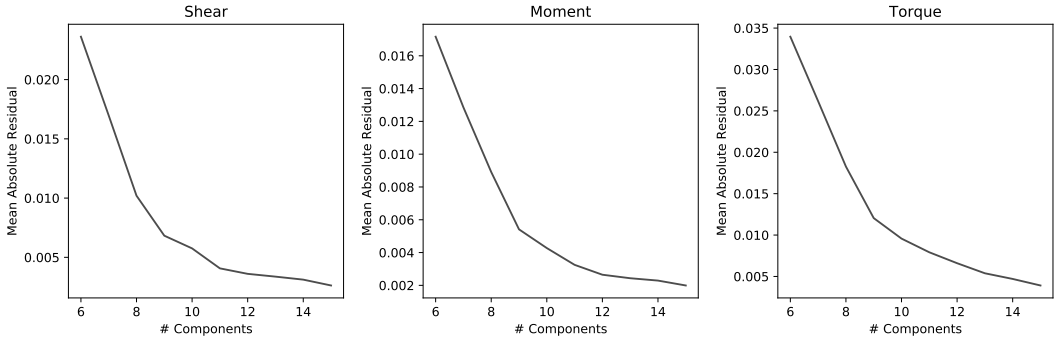

Figure D.1: Mean residuals of the enveloping loads as a function of the number of components used in a reduced dimensional PCA basis, separated for shear, moment and torque measurements.

D.4 Envelope Predictions

GP regression provides statistical predictions in terms of probability distributions. Predictions of q at a particular point, $\mathbf{z}^*$, are obtained by taking the expectation of this predictive distribution:

$$\hat{q} = \mathbb{E}[p(q^*|\mathbf{z}^*\mathcal{D}, \theta)] \tag{D.13}$$

where $p(q^*|\mathbf{z}^*, \mathcal{D}, \theta)$ is obtained as described in appendix B. Since the marginals of the predictive distribution are Gaussian (by definition), this is equivalent to the predictive mean, which can be calculated analytically.

Predictions of *loads* are obtained by retrieving predictions for $\mathbf{q} = [q_1, \dots q_R]$ from each of the R independent models, and multiplying this vector by the transpose of the PCA matrix, $\mathbf{P}$, to obtain a length M vector of predictions in the loads space, $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \hat{\mathbf{q}} \cdot \mathbf{P}^\intercal \tag{D.14}$$

This is performed for each of the K load cases to produce an $M \times K$ array of

vertically stacked responses, $\widehat{\mathbf{Y}}^*_{mk}$, for each load case:

$$\widehat{\mathbf{Y}}^* = \begin{bmatrix} \hat{\mathbf{q}_1} \cdot \mathbf{P}_1^\intercal \\ \vdots \\ \hat{\mathbf{q}_K} \cdot \mathbf{P}_K^\intercal \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{y}}_1^* \\ \vdots \\ \hat{\mathbf{y}}_K^* \end{bmatrix} \tag{D.15}$$

We finally obtain a prediction of the envelope, denoted $\overline{\mathbf{y}}^*$ by taking the coefficient-wise maximum across the load case axis (equivalent to the maximum of each column).

$$\overline{\mathbf{y}}^* = \max_{k=1\ldots K} |\widehat{\mathbf{Y}}_{mk}| \tag{D.16}$$

Appendix E

Basics of Hamiltonian Monte-Carlo

This appendix contains a brief introduction to the Hamiltonian Monte-Carlo simulation algorithm used during the construction of the robust surrogate models introduced in chapter 5.

E.1 Preliminaries

A comprehensive introduction to Hamiltonian Monte-Carlo is beyond the scope of this document, and we refer the curious reader to Betancourt [48] for an accessible conceptual introduction, or Gelman et al. [15, ch. 11-12] for a practical description of the algorithm.

A *Markov chain* is a sequence of points in D dimensional space, obtained from some random, initial point through repeated application of a *Markov transition* at each successive point. Markov transitions can be thought of as probability distributions, which – conditional on the current point – determine the probability of moving from that point to all other points in the space. By defining this probability distribution in such a way that it "preserves" some target distribution, one is able to obtain a sequence of points (a "chain") which is distributed according to the desired target distribution. This property is useful when it is either not possible or computationally efficient to sample from a target distribution directly.

Before addressing Hamiltonian Monte-Carlo simulation, it is useful to first briefly address the deficiencies of the Metropolis-Hastings algorithm, which is a more conceptually straightforward alternative.

E.1.1 Metropolis-Hastings

Perhaps the most well known procedure for constructing Markov transitions which preserve a target distribution is *Metroplis-Hastings* [25, 26]. The algorithm consists of two stages:

1. The **proposal** step induces a random perturbation to the current point.
2. The **correction** step then "rejects" any proposal which is too improbable (that is, if the probability of the new point given the target density is too low).

Metroplis-Hastings is simple and easy to implement, though the proposal-correction mechanism by which it operates is inefficient for high-dimensional spaces and/or

complicated target densities [48]: one can intuitively verify that if "probable" regions of the target density are small in comparison to the space to be explored, the majority of proposals will end up being rejected due to their small acceptance probability.

E.2 A Brief Conceptual Introduction to Hamiltonian Monte-Carlo Simulation

Hamiltonian Monte-Carlo simulation (HMC) [83, 84] is a technique which can be used to sample from higher-dimensional and/or complicated target densities in a considerably more efficient manner than the Metroplis-Hastings algorithm by exploiting the geometry of the target density. For *continuous* spaces, one way of accomplishing this is by defining a vector field aligned with the target density. This vector field can then be exploited to make *informed* proposals, which are able to rapidly traverse "likely" regions of the target density in order to reach unexplored regions of the space more efficiently.

Such a vector field is obtained by manipulating the gradients of the target density. While a thorough explanation in terms of differential geometry is complex and beyond the scope of this document, an intuitive way to conceptualise this procedure is to imagine an analogous (mathematically equivalent) physical system, in which each proposal in the probabilistic space is designed to mimic the motion of a fictitious particle through physical space in terms of *conservative dynamics*. Conservation of the momentum of this fictitious particle then ensures that the corresponding proposals in the probabilistic space remain "properly aligned" with the target density, in the sense that "motion" of the sampler through space is guided into regions of higher likelihood. HMC is sometimes referred to as *Hybrid* Monte-Carlo simulation as it includes deterministic mechanics as part of its random sampling.

E.2.1 Auxiliary Momentum Variable

Let θ represent a D dimensional point in the space, Θ, that we are interested in sampling from, and let $\mathcal{D}$ be some arbitrary data used to formulate the likelihood $p(\mathcal{D}|\theta)$, and corresponding posterior distribution $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)$. Since the algorithm only requires $p(\theta|\mathcal{D})$ to be known up to a multiplicative constant (i.e, we only require a distribution proportional to the target density), for the purposes of the algorithm, we can work with the likelihood directly (that is, assuming $p(\theta|\mathcal{D}) = p(\mathcal{D}|\theta)$).

HMC adds an auxiliary momentum variable, ϕ, for each dimension of Θ. We can thus conceptualise a joint distribution $p(\theta, \phi|\mathcal{D}) = p(\phi)p(\theta|\mathcal{D})$. HMC generates proposals from this joint distribution, with ϕ being "auxiliary" in the sense that we are only really interested in the samples of θ.

E.3 Algorithm

Each iteration of the HMC algorithm consists of three parts.

1. Randomly draw the momentum variable(s), ϕ.
2. Simultaneously update (θ, ϕ) over a total of L *leapfrog steps*.
3. Perform an accept/reject decision.

E.3.1 Drawing Momentum Variables

In most implementations, $p(\phi)$ is assigned a zero-mean multivariate normal distribution, with covariance matrix set to a diagonal "mass matrix", M, thus designating each auxiliary momentum variable as independent.

ϕ is consequently drawn randomly from, $\text{MVN}(\mathbf{0}, M)$, M being an appropriately defined mass matrix.

E.3.2 Simultaneous Update

Both θ and ϕ are changed together over a total of L iterations. These are referred to as "leapfrog" steps due to splitting the updates for the momentum variables into two half-updates.

In the following, let ε be a scaling parameter. At the beginning of each step the gradient of the log density of θ is used to perform a "half-update" of the auxiliary momentum variable, ϕ:

$$\phi \leftarrow \phi + \frac{1}{2}\varepsilon \frac{d \log p(\theta|\mathcal{D})}{d\theta} \tag{E.1}$$

Next, ϕ is used to update θ:

$$\theta \leftarrow \theta + \varepsilon M^{-1}\phi \tag{E.2}$$

Finally, another half-update of ϕ is performed:

$$\phi \leftarrow \phi + \frac{1}{2}\varepsilon \frac{d \log p(\theta|\mathcal{D})}{d\theta} \tag{E.3}$$

which completes a single leapfrog step.

This is performed a total of L times in order to jointly update θ and ϕ.

E.3.3 Accept/Reject Decision

Letting θ_{old} and ϕ_{old} be the values of θ and ϕ before the L leapfrog steps, and θ_{new} and ϕ_{new} their updated counterparts, we compute:

$$r = \frac{p(\theta_{\text{new}}|\mathcal{D})p(\phi_{\text{new}})}{p(\theta_{\text{old}}|\mathcal{D})p(\phi_{\text{old}})} \tag{E.4}$$

Then set:

$$\theta \leftarrow \begin{cases} \theta_{\text{new}} & \text{with probability} \quad \min(r, 1) \\ \theta_{\text{old}} & \text{otherwise} \end{cases} \tag{E.5}$$

In other words, we move to a new position (implicitly collecting a sample) with probability r, as defined by equation E.4.

E.4 Tuning Parameters

The HMC algorithm described in the preceding section can be tuned in four places: the mass matrix, M; the scaling factor, ε; and the number of leapfrog steps, L.

Gelman et al. [15, ch. 12] recommend setting the scale of the auxiliary momentum variables to a "crude estimate" of the scale of the target distribution, defaulting to setting M as the identity matrix of size D. Following this, tuning the scaling factor, ε, and number of leapfrog steps, L, such that $\varepsilon L = 1$, means that L steps of size ε fully traverses the (assumed scale of the) target density. The authors recommended a starting point of $\varepsilon = 0.01$ and $L = 10$.

E.4.1 Warmup and Adaptive Tuning

HMC is "optimally" efficient when the acceptance rate, r, is around 65% (see, for example, [85, section 5.4]). Since adaptively changing the HMC tuning parameters during sampling can produce transitions which do not necessarily preserve the target density, adaptively tuned sampling is preceded with a *warmup* phase during which the tuning parameters are adapted from their initial values to produce efficient sampling behaviour but no samples are actually collected.

Letting $\bar{r}$ be the average acceptance probability (computed from all of the iterations performed up until that point), it is recommended that ε and L are adapted such that $\bar{r} \approx 0.65$. If $\bar{r} \ll 0.65$, one should decrease ε and increase L, and if $\bar{r} \gg 0.65$, one should increase ε and decrease L, in both cases noting that the product $\varepsilon L = 1$

should be preserved.

E.5 Locally Adaptive HMC

Sample spaces for complicated models can feature geometry with regions of both low and high curvature. In such instances, it is desirable for the sampler to adapt to *local* properties of the target density, such that exploration is optimally efficient throughout the sample space.

One algorithm which accomplishes this is the *No-U-Turn Sampler* (NUTS), which is used throughout this document, as implemented in the probabilistic programming language *Stan* [46]. As NUTS is considerably more complicated than (non-locally-adapted) HMC, we refer the curious reader to the original work of Hoffman and Gelman [30] for a description of the algorithm.

Appendix F

Supplementary Material For Robust Surrogates Experiments

This appendix provides detailed documentation of the model fit procedures used in the experiments conducted in chapter 6.

F.1 MCMC Model Fit Procedure

The robust model is formulated as described in chapter 5. With reference to chapter 3 section 3.2, this yields the following probabilistic model:

$$\begin{aligned} \alpha &\sim \text{Normal}^{+}(0, 1) \\ \rho_i &\sim \text{InvGamma}(a_i, b_i) \quad i = 1 \dots D \\ \mathbf{y} &\sim \text{MVN}(\mathbf{0}, \mathbf{K}) \end{aligned} \tag{F.1}$$

where MVN is the multivariate normal distribution (recall chapter 3 for an explicit description of the likelihood function), and we reiterate that $\mathbf{K}$ is an $N \times N$ covariance matrix obtained by evaluating equation (6.1) pairwise between the rows of the $N \times D$ array of sample points, $\mathbf{X}$.

The hyperpriors for α and ρ are designated as described in chapter 5, with the parameters a_i and b_i representing the solution to the root finding problem described by the system of equations (5.5) in dimension i.

Given data, $\mathcal{D}$, composed of an $N \times D$ array of sample points, $\mathbf{X}$; and a length N vector of sample responses, $\mathbf{y}$, the MCMC method samples θ from an approximation to $p(\theta|\mathcal{D}, \pi_\theta)$ using Stan's implementation of NUTS, with settings as described in chapter 5.

A sketch of the pseudocode for the MCMC model fit procedure is documented in algorithm 4. We assume $\mathbf{X}$ and $\mathbf{y}$ have been normalised as described in section 5.4.1, such that the rows of $\mathbf{X}$ correspond to N points in D-dimensions $\mathbf{x} \in [0, 1]^D$; and $\mathbf{y}$ has *approximately* zero mean and unit standard deviation.

F.2 MLE Model Fit Procedure

The MLE model assumes flat hyperpriors. This results in a likelihood identical to the MVN PDF described by equation (3.5). We make use of Stan's automatic differentiation capabilities and optimisation routines for the MLE model, too.

Algorithm 4 MCMC Inference Pseudocode

Given: data: $N \times D$ array, $\mathbf{X}$; N vector $\mathbf{y}$
for $i = 1, \ldots, D$ **do**
    Compute Δ_i; the minimum spacing for column i of $\mathbf{X}$
    Compute a_i and b_i by solving (5.5), given Δ_i
end for
Initialise hyperprior π_θ, using a_i and b_i for $i = 1, \ldots, d$ ▷ see section (5.4)
Initialise sampler start points for θ ▷ see section (5.5.2)
Initialise sampler settings ▷ see section 5.5.1
while Not Satisfied **do**
    Run sampler
    Retrieve diagnostics: $\widehat{R}$, divergences, tree-depth saturations
    if Any failed diagnostics **then** ▷ see section 5.6
        Restart sampler with adapted settings
    else
        Satisfied ▷ the diagnostics indicate success
    end if
end while
Return: samples of θ from $p(\theta|\mathcal{D}, \pi_\theta)$

The probabilistic model, *implicitly* due to the absence of any explicit prior distribution for α or $\rho_1, \ldots, \rho_D$, is:

$$\begin{aligned} \alpha &\sim \text{Uniform}(0, \infty) \\ \rho_i &\sim \text{Uniform}(0, \infty) \quad i = 1 \ldots D \\ \mathbf{y} &\sim \text{MVN}(\mathbf{0}, \mathbf{K}) \end{aligned} \tag{F.2}$$

where $\mathbf{K}$ is (again) calculated via equation (6.1). The hyperpriors for α and ρ are factored out since all values of θ receive the same prior probability.

The MLE method attempts to summarise $p(\theta|\mathcal{D})$ by a single point, θ^*, assumed to be the point which globally maximises the log-likelihood:

$$\theta^* = \arg\max_{\theta \in \Theta} \log\left(\mathcal{L}(\theta|\mathcal{D})\right) \tag{F.3}$$

given $\log(\mathcal{L}(\theta|\mathcal{D}))$ corresponds to the MVN log-likelihood function (see, for example, section 3.2).

We approach the optimisation described by equation (F.3) using Newton's method as implemented in Stan, which uses automatic differentiation of the likelihood function to obtain the required derivatives. Convergence is determined by a change in the log-likelihood of less than $1e^{-10}$, and the maximum number of allowable number of iterations is set to 250. This limit was never reached during our experiments.

This procedure is repeated $R = 5$ times, and we keep the result with the highest log-likelihood. The first repeat is initialised from the same initial values as the MCMC algorithm (see section (5.5.2)), and the remaining 4 are initialised randomly.

The fit procedure for the MLE method is summarised by algorithm 5.

Algorithm 5 MLE Inference Pseudocode

Given: data: $N \times D$ array, $\mathbf{X}$; N vector $\mathbf{y}$
set $\theta^* = \texttt{NaN}$, $\mathcal{L}(\theta^*|\mathcal{D}) = -\infty$
for $r = 1, \ldots, R$ **do**
    Attempt to solve (F.3) using Newton's method, to obtain θ^*_{prop}
    if $\mathcal{L}(\theta^*_{\text{prop}}|\mathcal{D}) > \mathcal{L}(\theta^*|\mathcal{D})$ **then**
        set $\theta^* = \theta^*_{\text{prop}}$
    end if
end for
Return: θ^*

F.3 MLE-II Model Fit Procedure

The MLE-II model uses the same hyperpriors as the MCMC model, but summarises the posterior distribution $p(\theta|\mathcal{D}, \pi_\theta)$ by optimising the log-likelihood similarly to the MLE method.

This results in a probabilistic identical to that of the MCMC model:

$$\begin{aligned} \alpha &\sim \text{Normal}^+(0, 1) \\ \rho_i &\sim \text{InvGamma}(a_i, b_i) \quad i = 1 \ldots d \\ \mathbf{y} &\sim \text{MVN}(\mathbf{0}, \mathbf{K}) \end{aligned} \tag{F.4}$$

with hyperpriors for α and ρ designated identically to the MCMC model.

The objective function in this case is thus the following log *marginal* likelihood:

$$\log\left(\mathcal{L}(\theta|\mathcal{D},\pi_\theta)\right) = \log\left(p(\mathbf{y}|\mathbf{X},\theta)\right) + \log\left(p(\theta|\pi_\theta)\right) \tag{F.5}$$

where $\log(p(\mathbf{y}|\mathbf{X},\theta))$ is given by the log of the MVN PDF with zero mean and covariance matrix $\mathbf{K}$, and $p(\theta|\pi_\theta)$ is the log of the joint probability of α and $\rho_1,\ldots,\rho_d$ as determined by their respective prior distributions described by equation (F.4).

We seek:

$$\theta^* = \arg\max_{\theta\in\Theta} \log\left(\mathcal{L}(\theta|\mathcal{D},\pi_\theta)\right) \tag{F.6}$$

and solve the optimisation problem described by equation (F.6) identically to the MLE case.

The fit procedure for the MLE method is summarised by algorithm 6.

Algorithm 6 MLE-II Inference Pseudocode

Given: data: $N \times D$ array, $\mathbf{X}$; N vector $\mathbf{y}$
for $i = 1,\ldots,D$ **do**
    Compute Δ_i; the minimum spacing for column i of $\mathbf{X}$
    Compute a_i and b_i by solving (5.5), given Δ_i
end for
Initialise hyperprior π_θ, using a_i and b_i for $i = 1,\ldots,D$
set $\theta^* = \texttt{NaN}$, $\mathcal{L}(\theta^*|\mathcal{D},\pi_\theta) = -\infty$
for $r = 1,\ldots,R$ **do**
    Attempt to solve (F.6) using Newton's method, to obtain θ^*_{prop}
    if $\mathcal{L}(\theta^*_{\text{prop}}|\mathcal{D},\pi_\theta) > \mathcal{L}(\theta^*|\mathcal{D},\pi_\theta)$ **then**
        set $\theta^* = \theta^*_{\text{prop}}$
    end if
end for
Return: θ^*

F.4 Test Functions

3D visualisations of the 2D test functions (Branin, mixture of cosines, and 6-hump camel) are shown in figures (F.1), (F.2) and (F.3), respectively. To resolve the detail

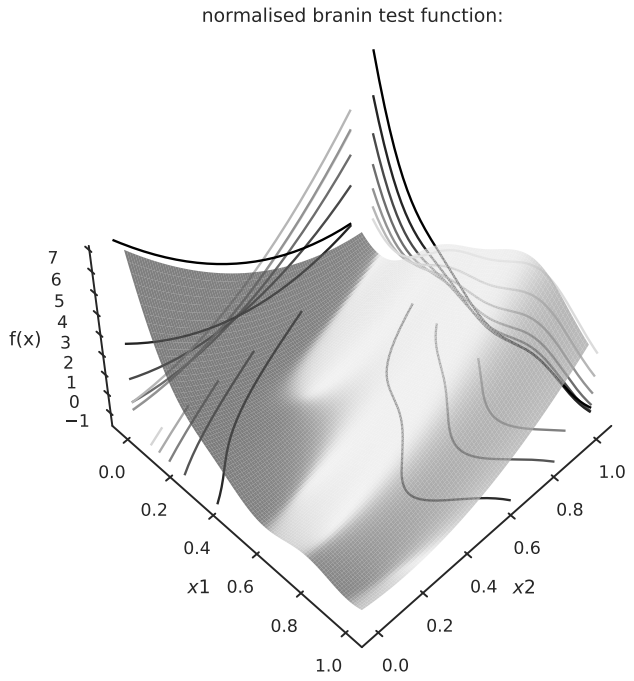

Figure F.1: Surface and contour plot for the Branin test function.

at the bottom of the central basin region in the case of the 6-hump camel function, an additional plot of just this region is shown in figure (F.4).

Mathematical descriptions of the mixture of cosines and 4D sphere test functions are provided in the following sections.

F.4.1 Mixture of Cosines

Our experiments use $D = 2$. See also figure (F.2).

$$f(\mathbf{x}) = 0.1 \sum_{i=1}^{D} \cos 5\pi x_i - \sum_{i=1}^{D} x_i^2 \tag{F.7}$$

F.4.2 4D Hypersphere

Our experiments use $D = 4$.

$$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 \tag{F.8}$$

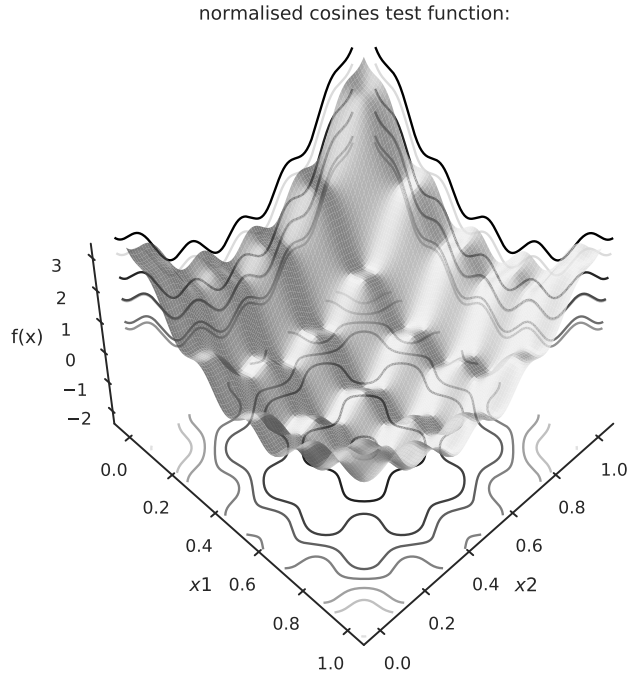

Figure F.2: Surface and contour plot for the mixture of cosines test function.

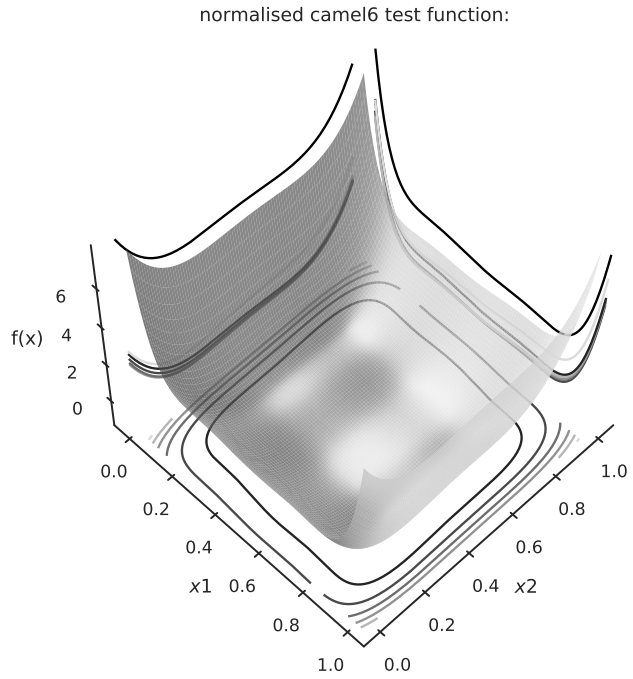

Figure F.3: Surface and contour plot for the 6-hump camel test function. As the geometry of the basin region is difficult to resolve on this scale, an additional plot of the basin is shown in figure (F.4).

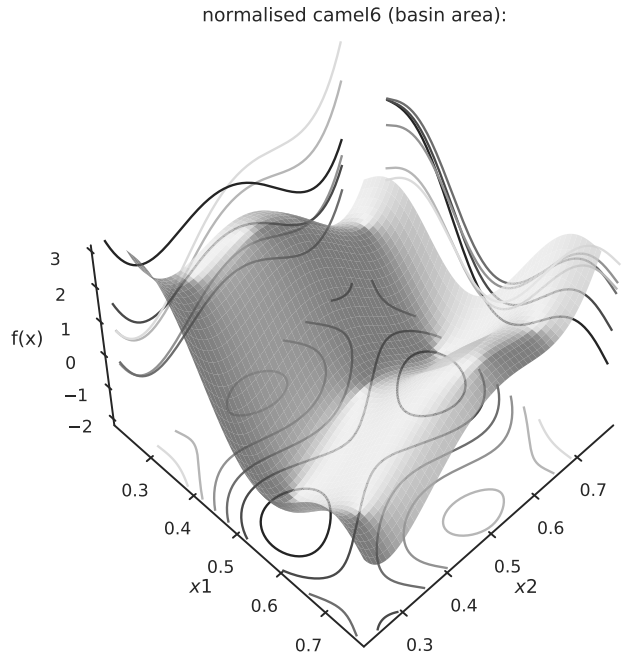

Figure F.4: Surface and contour plot for the 6-hump camel test function basin region.

Appendix G

Adaptive Data Acquisition and Bayesian Optimisation

This appendix provides a brief introduction to adaptive data acquisition and Bayesian optimisation, with a short review of extensions useful to industrial modelling applications.

G.1 Adaptive Data Acquisition

We introduce Bayesian Optimisation (BO) by first formalising adaptive data acquisition as follows.

Let $\mathcal{D}_t \equiv \{\mathbf{X}, \mathbf{y}\}_t$ be the data available at time, t, which consists of a set of N sample points, $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]$, and a corresponding vector of N scalar responses, $\mathbf{y}$, obtained by evaluating a simulator, f, at each of the points in $\mathbf{X}$. We presume at each t, $\mathcal{D}_t$ is used to construct a GP emulator.

Using this emulator, we can compute the distribution of a predicted quantity, y^*, at any arbitrary point in the index, $\mathbf{x}^*$, conditional on the observed data and model hyperparameters. We denote the predictive distribution by $p(y^*|\mathbf{x}^*, \mathcal{D}_t, \theta)$. For GP models, this distribution is fully defined by a predictive mean, $\mathbf{m}$, and covariance matrix, $\mathbf{K}$, obtained as described in appendix 2.

An adaptive experimental design seeks to provide a strategy for selecting one (or more) new points, $\mathbf{x}_{t+1}$, which along with the corresponding responses, $\mathbf{y}_{t+1}$, are used to build a new emulator at time $t+1$. This typically involves maximising some function of the predictive distribution.

G.2 Acquisition Strategies

Functions of the predictive distribution which are used to select points in the index at which to acquire new samples are referred to as *acquisition functions*. Let a be an arbitrary acquisition function. We write a as conditioned on a specific predictive distribution as $a(\mathbf{x}^*|y^*, \mathcal{D}_t, \theta)$, noting that since y^* is conditional on the data available at time t, $\mathcal{D}_t$, and model hyperparameters, θ, then the acquisition function must be, too. To select the next point, one maximises (or minimises) $a(\mathbf{x}^*|y^*, \mathcal{D}_t, \theta)$ with respect to $\mathbf{x}$:

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}^* \in \mathcal{X}} a(\mathbf{x}^*|y^*, \mathcal{D}_t, \theta) \tag{G.1}$$

An acquisition function can be consequently thought of as some means of quantifying the "usefulness" of all the points $\mathbf{x} \in \mathcal{X}$, and hence provides some mechanism of reasoning with their relative utility. Identifying the global optimum in equation (G.1) can be challenging to accomplish *efficiently* as acquisition functions tend to be multi-modal. They are, however, usually relatively inexpensive to compute and permit analytical derivatives, such that multi-start Newton or quasi-Newton methods are effective in low to moderate numbers of dimensions.

It is emphasised that how realistically an acquisition function represents the "usefulness" it defines (and consequently, how effective it is in that context) is conditional on how well the GP model on which it is conditional represents predictive uncertainty across the index. Emulators constructed using robust methods, such as those described by chapter 5, are thus highly beneficial, despite implying extra computational effort is required as a result of marginalisation.

For GP models fit using MCMC, the objective function in equation (G.1) should be marginalised over the posterior distribution of the hyperparameters. Assuming S samples of θ have been drawn from $p(\theta|\mathcal{D}_t, \pi_\theta)$ (where π_θ is the prior distribution for θ), the marginalised acquisition function is:

$$a(\mathbf{x}^*|y^*, \mathcal{D}_t) = \frac{1}{S}\sum_{s=1}^{S} a(\mathbf{x}^*|y^*, \mathcal{D}_t, \theta_s) \tag{G.2}$$

which can be inserted directly into equation (G.1) in place of $a(\mathbf{x}^*|y^*, \mathcal{D}_t, \theta)$. From here we will omit explicit statements of conditionality on the model hyperparameters, removing θ from $a(\mathbf{x}^*|y^*, \mathcal{D}_t, \theta_t)$ in the interest of readability, and consider it implicit that if MCMC is used then predictions are marginalised over $p(\theta|\mathcal{D}_t, \pi_\theta)$ as described by equation (G.2). Similarly, we will also drop the conditional variable $\mathcal{D}_t$, assuming it is implicit that the predictive distribution, $p(y^*|\mathbf{x}^*)$ obtained at time t makes use of all of the available data at that time unless stated otherwise.

G.2.1 Pure Exploration

Different acquisition functions formalise utility in different ways. This can range from relatively simple measures to complex functions of the GP posterior. Perhaps the most simple formulation of adaptive experimental design is to select the next point at the location of maximum posterior variance:

$$a(\mathbf{x}) = \mathbb{V}[p(y^*|\mathbf{x}^*] \tag{G.3}$$

which is otherwise known as *Pure Exploration* (PX); the acquisition function simply acquires data at the point of maximum predictive uncertainty.

Strictly speaking, PX does not require *any* responses from the simulator we are seeking to optimise using BO, as the posterior predictive variance is a function of only the separation between data points and the model hyperparameters (see chapter 2). However, feedback from the target function via observed data is still important, as additional datapoints constitutes extra "evidence" in identifying suitable values of the hyperparameters.

PX is cheap to compute and can be analytically differentiated, making it inexpensive to optimise.

G.2.2 Other Global Design Criteria

Adaptive experimental design for Gaussian process emulators was studied by Busby [86].

G.3 Bayesian Optimisation

Global design and pure exploration are inefficient if one requires accuracy in only a specific region of the input space: a canonical example of such a requirement is when using the surrogate to perform (global) optimisation. In these cases, use of an acquisition function to adaptively construct a surrogate model which is then used to predict the minimum of the original (expensive) function is known as Bayesian

Optimisation. Readers are directed to Shahriari et al. [60] for an up-to-date review.

G.3.1 Expected Improvement

A wide range of acquisition functions exist for performing BO.

Moćkus et al. [87] derived the now-ubiquitous *Expected Improvement* (EI) acquisition function based on a quantification of the expected increase in the observed value of the random variable defined by the GP posterior marginals relative to an incumbent value, y^*. The incumbent is defined either as the largest observation so far, i.e. $y^* = \max(\mathbf{y})$; or as the maximum of the predictive mean of the GP, i.e. $y^* = \arg\max_{\mathbf{x}\in\mathcal{X}} \mu(\mathbf{x}|\mathcal{D})$.

Given y^*, EI is defined as:

$$EI(\mathbf{x}) = (y^* - \mu(\mathbf{x}))\,\Phi(\gamma) + \sigma(\mathbf{x})\phi(\gamma) \tag{G.4}$$

where:

$$\gamma = \frac{y^* - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \tag{G.5}$$

$\Phi(\cdot)$ and $\phi(\cdot)$ are the normal (Gaussian) cumulative distribution function and probability density function, respectively.

EI is popular as it is simple to implement and offers reliable theoretical guarantees of convergence under relatively mild assumptions, crucially including uncertainty in the GP hyperparameters [88, 89], which is often the case in practice.

G.3.2 Predictive Entropy Search

Hernández-Lobato et al. [61] developed predictive entropy search (PES); an acquisition function based on maximising the (estimated) information gain about the location of the global optimum associated with retrieving a sample at a particular

point.

Relative to EI, PES has a stronger preference for exploration, giving it a reduced tendency to be stuck inside local minima during early iterations, and state-of-the-art empirical performance when compared to other acquisition functions on a variety of benchmarks (see also favourable comparisons in [66]).

PES also offers the possibility of being utilised for purposes other than optimisation. By maximising the information gain with respect to, for example, regions of high curvature, one could construct an acquisition function which maximises the information retrieval at regions with high absolute gradients in the input space.

Unfortunately, PES suffers from some practical difficulties which make application in industrial production environments tricky. First, the algorithm requires provision and management of high (up to fourth) order derivatives of the GP kernel. This is straightforward for simple kernels such as the exponentiated quadratic, but is cumbersome for more complicated kernels. This could be remedied by using either a spectral representation of the GP kernel, such as that deployed by [90] for similar purposes, or the use of automatic differentiation to retrieve the high order derivatives of the kernel without programming them analytically or resorting to finite difference schemes.

Second, when predicting potential locations for the function minimum, the PES algorithm makes several (necessary) assumptions about the derivatives at the minimum; namely that the Hessian matrix is negative definite and the first derivative is zero. These assumptions are valid if the function minimum lies well within the bounds of the search space, but are sometimes *invalid* if the minimum lies on any of the boundaries of the input space. Should this occur, the expectation propagation step (see [91]) of the algorithm may not converge, meaning the acquisition function cannot be evaluated. In our experiments, this occurs frequently enough that the usability of the algorithm is compromised. This is perhaps less of a concern in applications in which the search space can be arbitrarily extended.

G.4 Batch Acquisition

In many cases it is convenient to sample a batch of points at once. This is especially the case if the original function can be run in parallel, or if the function requires time to be set up, such as if data acquisition involves a physical experiment. In such situations, it is considerably more convenient to retrieve a batch of samples at a time.

Batch acquisition has been explored by several authors in the Bayesian Optimisation literature. González et al. [92] used a local penalisation method to emulate the effect of an observation on the posterior distribution to provide a sequentially greedy strategy for batch allocation using EI. This approach is fast and easy to implement, but cannot guarantee an optimal solution to the batch design due to the greedy nature of the acquisition function.

González et al. developed GLASSES [93], a non-greedy approach to batch allocation, which seeks to approximate an "ideal" look-ahead function with a surrogate. GLASSES seeks to minimise an approximation to a *long sight loss function* (see [94]), which involves successive optimisation and marginalisation of a sequential acquisition function. Lam et al. [66] approached a similar problem using a dynamic programming solution based on rollout (see [95, ch.4]), attempting to maximise the cumulative utility of the batch given a fixed budget of simulation points, marginalising over uncertainty in "virtual" observations using quadrature.

Shah and Ghahramani [68] extended PES to allocate a batch of points maximising the *joint* information gain about possible locations for the function minimum. Batch PES is non-greedy but suffers from the same implementation issues as single-point PES, namely provision of high order kernel derivatives and instability of the acquisition function when the function optimum resides on or around the support of the target function.

G.5 Multitask Acquisition

For applications in which different objectives are not necessarily conflicting, we refer to multi-task optimisation. Here, one seeks to simultaneously search for the optima of two or more objective functions which share an input space. A simple means of accomplishing this is to take the expectation of the acquisition function across the different tasks. Let $a_t(\mathbf{x})$ be the acquisition function for task t, and presume T tasks. A generic multi-task acquisition function, $\alpha(\mathbf{x})$, can be defined as:

$$\alpha(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} a_t(\mathbf{x}) \tag{G.6}$$

Weights can optionally be added to the summands in equation (G.6) to force a preference for certain tasks. It is important that the tasks should be expressed in the same or similarly scaled units.

G.6 Multiobjective Acquisition

When faced with more than one objective, one can consider multiobjective acquisition. Objectives can come from either multiple independent GP models, or a single, vector-output model. The acquisition function in this case seeks to identify the pareto optimal set of points.

Emmerich [96] proposed *Expected Hypervolume Improvement* (EHVI) as a means of applying Bayesian Optimisation to a multiobjective setting. A more computationally efficient version of the algorithm was later developed by Hupkens et al. [97].

Hernández-Lobato et al. [67] extended PES to the multiobjective setting, using a reformulation of the PES acquisition function to describe utility in terms of the information gain about the pareto set.

Bibliography

[1] Pranab K Muhuri, Amit K Shukla, and Ajith Abraham. Industry 4.0: A bibliometric analysis and detailed overview. *Engineering applications of artificial intelligence*, 78:218–235, 2019.

[2] Lucian Iorga, Vincent Malmedy, Olivia Stodieck, Joe Loxham, and Simon Coggon. Preliminary sizing optimisation of aircraft structures - industrial challenges and practices. In *6th Aircraft Structural Design Conference*. RAeS, October 2018.

[3] Gustavo Krupa, Joe Loxham, Timoleon Kipouros, Simon Coggon, Sanjiv Sharma, and A.M. Savill. External aircraft load variability due to correlated structural design uncertainties. In *Aerodynamics Tools And Methods in Aircraft Design*. RAeS, October 2019.

[4] Anthony O'Hagan. Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering & System Safety*, 91(10-11):1290–1300, 2006.

[5] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

[6] Michael Scheuerer, Robert Schaback, and Martin Schlather. Interpolation of spatial data–a stochastic or a deterministic problem? *European Journal of Applied Mathematics*, 24(4):601–629, 2013.

[7] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[8] Andre G Journel and ME Rossi. When do we need a trend model in kriging? *Mathematical Geology*, 21(7):715–739, 1989.

[9] Juho Piironen and Aki Vehtari. Projection predictive model selection for gaussian processes. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

[10] David JC MacKay et al. Bayesian nonlinear modeling for the prediction competition. *ASHRAE transactions*, 100(2):1053–1062, 1994.

[11] RJ Adler. *The Geometry of Random Fields*. John Wiley, 1981.

[12] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667, 2006.

[13] Andrew Gordon Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015.

[14] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[15] Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[16] Felipe AC Viana, Timothy W Simpson, Vladimir Balabanov, and Vasilli Toropov. Special section on multidisciplinary design optimization: metamodeling in multidisciplinary design optimization: how far have we really come? *AIAA journal*, 52(4):670–690, 2014.

[17] Alexander IJ Forrester and Andy J Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1-3):50–79, 2009.

[18] Bertrand Iooss and Paul Lemaître. A review on global sensitivity analysis methods. In *Uncertainty management in simulation-optimization of complex systems*, pages 101–122. Springer, 2015.

[19] Ercan Solak, Roderick Murray-Smith, William E Leithead, Douglas J Leith, and Carl E Rasmussen. Derivative observations in gaussian process models of

dynamic systems. In *Advances in neural information processing systems*, pages 1057–1064, 2003.

[20] Jaakko Riihimäki and Aki Vehtari. Gaussian processes with monotonicity information. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 645–652, 2010.

[21] Stan Development Team. Stan modeling language users guide and reference manual version 2.18.0, 2018. URL `https://mc-stan.org/docs/2_18/stan-users-guide` last accessed on 10-04-2019.

[22] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[23] S Sundararajan and S Sathiya Keerthi. Predictive app roaches for choosing hyperparameters in gaussian processes. In *Advances in neural information processing systems*, pages 631–637, 2000.

[24] Radford M Neal. *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada, 1993.

[25] N Metroplis, AW Rosenbluth, MN Rosenbluth, and AH Teller. Equation of state calculations by fast computing machines. *J. chem. Phys.*, 21(6):1986–1092, 1953.

[26] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[27] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *Readings in computer vision*, pages 564–584. Elsevier, 1987.

[28] Radford M. Neal. Slice sampling. *Ann. Statist.*, 31(3):705–767, 06 2003. doi: 10.1214/aos/1056562461. URL `https://doi.org/10.1214/aos/1056562461`.

[29] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[30] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[31] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[32] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112 (518):859–877, 2017.

[33] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

[34] Yarin Gal, Mark Van Der Wilk, and Carl Edward Rasmussen. Distributed variational inference in sparse gaussian process regression and latent variable models. In *Advances in neural information processing systems*, pages 3257–3265, 2014.

[35] Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Yes, but did it work?: Evaluating variational inference. *arXiv preprint arXiv:1802.02538*, 2018.

[36] Jay D Martin and Timothy W Simpson. Use of kriging models to approximate deterministic computer models. *AIAA journal*, 43(4):853–863, 2005.

[37] Hirotugu Akaike. A new look at the statistical model identification. In *Selected Papers of Hirotugu Akaike*, pages 215–222. Springer, 1974.

[38] François Bachoc. Cross validation and maximum likelihood estimations of hyper-parameters of gaussian processes with model misspecification. *Computational Statistics & Data Analysis*, 66:55–69, 2013.

[39] Jonathan Cooper, H Khodaparast, Sergio Ricci, G Georgiou, Gareth Vio,

L Trawaglini, and P Denmer. Rapid prediction of worst case gust loads. In *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, page 2040, 2012.

[40] Max D Morris and Toby J Mitchell. Exploratory designs for computational experiments. *Journal of statistical planning and inference*, 43(3):381–402, 1995.

[41] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[42] Irene Tartaruga, Pia Sartor, Jonathan E Cooper, Simon Coggon, and Yves Lemmens. Efficient prediction and uncertainty propagation of correlated loads. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1847, 2015.

[43] Irene Tartaruga, Jonathan E Cooper, Pia Sartor, Mark H Lowenberg, and Yves Lemmens. Geometrical based method for the uncertainty quantification of correlated aircraft loads. *Journal of Aeroelasticity and Structural Dynamics*, 4(1), 2016.

[44] Michael Betancourt. Robust gaussian processes in stan. `https://betanalpha.github.io/assets/case_studies/gp_part3/part3.html`, 2017.

[45] Jorge J Moré, Burton S Garbow, and Kenneth E Hillstrom. User guide for minpack-1. Technical report, CM-P00068642, 1980.

[46] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

[47] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of mcmc. *arXiv preprint arXiv:1903.08008*, 2019.

[48] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[49] Radford M Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003.

[50] Sonja Surjanovic and Derek Bingham. Virual library of simulation experiments, 2013. URL `https://www.sfu.ca/~ssurjano/index.html` last accessed on 21-08-2019.

[51] Laurence Charles Ward Dixon. The global optimization problem. an introduction. *Toward global optimization*, 2:1–15, 1978.

[52] M Montaz Ali, Charoenchai Khompatraporn, and Zelda B Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of global optimization*, 31(4):635–672, 2005.

[53] Marcin Molga and Czesław Smutnicki. Test functions for optimization needs. *Test functions for optimization needs*, 101, 2005.

[54] Jerome H Friedman et al. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

[55] Einat Neumann Ben-Ari and David M Steinberg. Modeling data from computer experiments: an empirical comparison of kriging with mars and projection pursuit regression. *Quality Engineering*, 19(4):327–338, 2007.

[56] William V Harper and Sumant K Gupta. *Sensitivity/uncertainty analysis of a borehole scenario comparing Latin hypercube sampling and deterministic sensitivity approaches*. Office of Nuclear Waste Isolation, Battelle Memorial Institute, 1983.

[57] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[58] Ilya M Sobol. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics*, 16(5): 236–242, 1976.

[59] Sergei Kucherenko, Daniel Albrecht, and Andrea Saltelli. Exploring multidimensional spaces: A comparison of latin hypercube and quasi monte carlo sampling techniques. *arXiv preprint arXiv:1505.02350*, 2015.

[60] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[61] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems*, pages 918–926, 2014.

[62] Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[63] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st conference on Visualization'90*, pages 361–378. IEEE Computer Society Press, 1990.

[64] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. *arXiv preprint arXiv:1703.01968*, 2017.

[65] Binxin Ru, Mark McLeod, Diego Granziol, and Michael A Osborne. Fast information-theoretic bayesian optimisation. *arXiv preprint arXiv:1711.00673*, 2017.

[66] Remi Lam, Karen Willcox, and David H. Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 883–891. Curran Associates, Inc., 2016.

[67] Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pages 1492–1501, 2016.

[68] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for

batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, pages 3330–3338, 2015.

[69] Hyoung-Moon Kim, Bani K Mallick, and CC Holmes. Analyzing nonstationary spatial data using piecewise gaussian processes. *Journal of the American Statistical Association*, 100(470):653–668, 2005.

[70] Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. Generating random correlation matrices based on vines and extended onion method. *Journal of multivariate analysis*, 100(9):1989–2001, 2009.

[71] Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in neural information processing systems*, pages 329–336, 2004.

[72] Chunyi Wang and Radford M Neal. Gaussian process regression with heteroscedastic or non-gaussian residuals. *arXiv preprint arXiv:1212.6246*, 2012.

[73] Michael Betancourt. Probability theory for scientists and engineers, October 2018. URL `https://betanalpha.github.io/assets/case_studies/probability_theory` last accessed on 10-04-2019.

[74] Michael Betancourt. Conditional probability theory for scientists and engineers, October 2018. URL `https://betanalpha.github.io/assets/case_studies/conditional_probability_theory` last accessed on 10-04-2019.

[75] Stan Development Team. Stan functions reference version 2.19.0, 2018. URL `https://mc-stan.org/docs/2_19/functions-reference` last accessed on 10-04-2019.

[76] Morris Eaton. *Multivariate statistics : a vector space approach*. Wiley, New York, 1983. ISBN 0-471-02776-6.

[77] David K Duvenaud, Hannes Nickisch, and Carl E Rasmussen. Additive gaussian processes. In *Advances in neural information processing systems*, pages 226–234, 2011.

[78] Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. Input warping for

bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, pages 1674–1682, 2014.

[79] Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, pages 393–400. ACM, 2007.

[80] David John James Toal and Andy J Keane. Non-stationary kriging for design optimization. *Engineering Optimization*, 44(6):741–765, 2012.

[81] J Bernardo, J Berger, A Dawid, and A Smith. Some bayesian numerical analysis. *Bayesian statistics*, 4:345–363, 1992.

[82] Richard Dwight and Zhong-Hua Han. Efficient uncertainty quantification using gradient-enhanced kriging. In *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 17th AIAA/ASME/AHS Adaptive Structures Conference 11th AIAA No*, page 2276, 2009.

[83] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[84] Radford M Neal. An improved acceptance procedure for the hybrid monte carlo algorithm. *Journal of Computational Physics*, 111(1):194–203, 1994.

[85] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[86] Daniel Busby. Hierarchical adaptive experimental design for gaussian process emulators. *Reliability Engineering & System Safety*, 94(7):1183–1193, 2009.

[87] J Moćkus, V Tiesis, and A Źilinskas. The application of bayesian methods for seeking the extremum. vol. 2, 1978.

[88] Adam D Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(Oct):2879–2904, 2011.

[89] Ziyu Wang and Nando de Freitas. Theoretical analysis of bayesian opti-

misation with unknown gaussian process hyper-parameters. *arXiv preprint arXiv:1406.7758*, 2014.

[90] Anqi Wu, Mikio C Aoi, and Jonathan W Pillow. Exploiting gradients and hessians in bayesian optimization and bayesian quadrature. *arXiv preprint arXiv:1704.00060*, 2017.

[91] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[92] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial Intelligence and Statistics*, pages 648–657, 2016.

[93] Javier González, Michael Osborne, and Neil Lawrence. Glasses: Relieving the myopia of bayesian optimisation. In *Artificial Intelligence and Statistics*, pages 790–799, 2016.

[94] M Osborne. *Bayesian Gaussian processes for sequential prediction, optimisation and quadrature*. PhD thesis, University of Oxford, 2010.

[95] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 2005.

[96] Michael Emmerich. Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels. *Dissertation, LS11, FB Informatik, Universität Dortmund, Germany*, 2005.

[97] Iris Hupkens, Michael Emmerich, and André Deutz. Faster computation of expected hypervolume improvement. *arXiv preprint arXiv:1408.7114*, 2014.