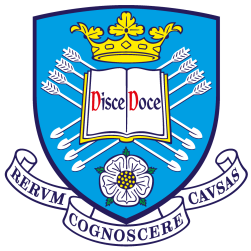


The University Of Sheffield.

A framework for combining model calibration with model-based optimization in virtual engineering design workflows

Submitted September 2020, in partial fulfillment of
the conditions for the award of the degree **PhD - Automatic Control and Systems Engineering.**

Oliver Jones
160120664

Supervised by Professor Robin Purshouse and Professor Jeremy Oakley

Department of Automatic Control and Systems Engineering
The University of Sheffield

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____
Date _____ / _____ / _____

To my family and all my friends

Thank you for helping me see this adventure through to the end

"It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers." - Patrick Rothfuss, The Wise Man's Fear

Acknowledgements

I owe my deepest gratitude to my supervisor Professor Robin Purshouse. He has been supporting me ever since I came to Sheffield University. Without his continuous guidance, enthusiasm and optimism concerning this work this project would not be where it is today. I also express my warmest gratitude to my other supervisor Professor Jeremy Oakley who has supported me over these last 4 years. He has always been willing to lend a hand and point me in the right direction when I am lost. Within the University of Sheffield, I would also like to thank Mathew Ham for his administrative legwork and Darren Fox for his equipment procurement.

To my friends, Zak and Shiv, thank you for always being there day or night whether it was for a simple sanity check or proofreading a document. To Daniel, Harry, Buket and Abidemi along with my other friends from university thank you for making this experience such an adventure and for always being willing to come over and say hi.

Finally, to my family. To my brothers and sister, thank you for your continued support whether it was helping with work or even just checking up on me. To my parents, you have always being there for me and never doubting I would succeed, thank you.

Abstract

In recent years, the development of complex engineering products has seen a movement towards increasing levels of virtualisation using expensive black-box simulations. One of the main factors driving this trend is the rapid increase in available computational resources. As computational capabilities are further developed, projects which used to be infeasible are now possible.

When using a virtual engineering design process, once the structure of the simulation model has been built, there is a need to perform both calibration and optimization in order to ensure that the resulting outputs presented to a decision maker correctly represent the optimal solutions. Both of these stages require the use of model evaluations to determine the efficacy of new parameterizations and designs. Such usage becomes a problem when there is only a limited budget of evaluations available within the design process for both stages. This problem is reinforced further by the current practice of considering the two stages as separate problems where there is only a limited transfer of knowledge between them, rather than a linked process.

The question that is posed within this research is whether there would be any benefits to adopting a linked approach to the calibration and optimization of expensive multi-objective design problems.

In order to determine an answer to this question, it is first essential to set out a mathematical formulation for the joint problem of calibration and optimisation. In order to assess any newly developed methods, it is necessary to devise a set of benchmark problems that contain both model parameters and control inputs that are required to be determined. This is achieved through the adaptation of pre-existing problems from the optimization literature as well as the development of a new component that fits within the Walking Fish Group (WFG) framework. A new alternating combined methodology is developed with the aim of gaining information within more relevant areas of the search space to improve the efficiency of the evaluations used. This new alternating method is further expanded to incorporate surrogates with the aim of improving knowledge sharing between the stages of model calibration and optimization. It is found that the use of the new alternating method can improve the final parameter sets obtained by the calibration, when compared to the classical approach. The extended alternating approach also offers superior calibration, in addition to achieving faster improvement in convergence of the optimiser to the true Pareto front of optimal designs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis examines the design stages of model calibration and optimization in order to determine if their overall efficiency can be increased through considering them as a combined problem. The research fields for both these stages are very active and their presence can be seen throughout both academia and industry. While this is the case, currently the fields of model calibration and optimization are treated almost exclusively as separate stages. Within this work the focus lies upon exploring the possibility of crossover and combination between them. Before continuing it is useful to lay out precisely how this work fits into the larger engineering design process.

In its most broad sense the engineering design process is the series of steps taken by an engineer to create a functional product. During the process of engineering design, modelling is one of the key stages that needs to be considered. There are two main methods used to perform modelling: prototyping and computer simulation. Numerous factors influence the decision as to which of these two methods are preferable. Such factors include the manufacturing costs of producing the simulation or prototype and how long they take to make. When prototyping it can be difficult to change the parameters of the product. For instance, when designing a plane it would not be practical to build new wings of each possible shape that was to be tested due to the expense and possible safety issues. The alternative to this is the production of a computer simulation which aids in supporting the engineering design. The advantage of producing a computer model is that once it is constructed, alterations can be applied to it at relatively low expense. Despite the adaptability that can be obtained through the use of a computer model, they suffer from the issue of often being expensive to build as well as never being fully trustworthy. While in many cases, the minor differences between any simulated results and reality would not matter, there are exceptions such as when there is a risk of harm or damage occurring. In these cases, it is important to ensure that the product will act as expected, so the use of prototypes is advisable. A schematic of the virtual engineering design workflow can be seen in Figure 1.1.

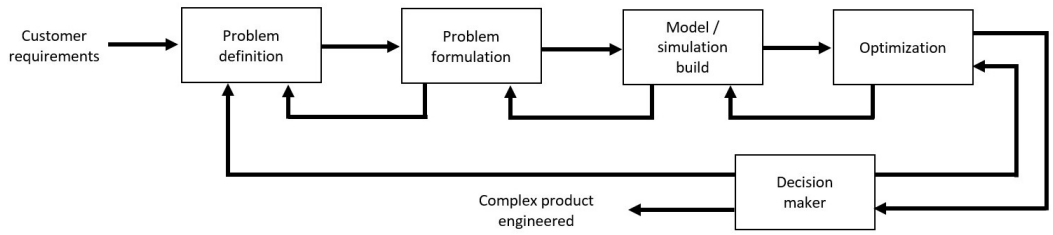


Figure 1.1: A schematic of the virtual engineering workflow

The steps within the workflow are:

- Customer requirements - Lay out a list of requirements provided by the customer detailing what they want from the final product
- Problem definition – Produce a textual description of the problem, including where it fits within the process of the organization
- Problem formulation – Develop a mathematical formulation of the problem, including objectives, constraints, design variables and parameters
- Model / simulation build – Construct the model presented in the problem formulation. This construction consists of two stages, the creation of a model structure and the calibration of internal parameters.
- Optimization – Acquire the best possible solutions from the model
- Decision maker – Ask the decision maker whether the found solutions are acceptable
- Complex product engineered – Return the finalised product to the customer.

The two stages of calibration (part of the model build) and optimization which are of interest for this research both require the use of function evaluations. In the case of model calibration, the function evaluations are used for the production and improvement of the system model through alteration of the internal parameters in order to ensure that the system outputs represent reality as closely as possible. Optimization utilises the function evaluations for selecting inputs to the system that provide the most desirable outputs. When working on real world problems it is often the case that only a set number of function evaluations are available for use throughout the entire design process. For simple problems this does not pose an issue as the number of evaluations available often lies above the tens of thousands. When the system is more complex however the cost of the function evaluations can become prohibitive. For instance, if a single evaluation were to take an hour of computer time to run, then it may only be possible to perform a couple of hundred evaluations over the course of an entire project. The focus of this work is on when such instances occur. This means that while within both research fields there are a multitude of methods for performing their respective steps, a large proportion of these are not suitable when only a small evaluation budget is present.

Within industry, especially within some larger companies, it is common practice for there to be separate departments handling each of the calibration and optimization stages. These departments will contain specialists that are proficient at their field but may only possess superficial knowledge of the other. Due to this separation the majority of information gathered during the calibration stage

is thrown away with only a minimal transfer of knowledge occurring when the parameters are passed to those working on optimization.

In this work the broad perspective is maintained when considering the two stages with a focus on considering them as a single combined problem. A mathematical framework for the combined problem is produced in order to allow for better understanding of how calibration and optimization can be linked. The new frameworks viability is demonstrated by setting a real-world injection moulding problem within it. Possible assessment criteria for determining the performance of the combined problem are discussed. In order to allow for benchmarking a new set of test problems based on a selection drawn from the optimization literature were developed with both control inputs and parameters present. A novel approach to solving the combined problem through the use of an alternating methodology is detained and compared to the classical approach. This novel method was further developed for use with a limited evaluation budget with the aim of solving expensive problems.

The remainder of this chapter begins, in Section 1.1, by presenting the motivation for this research. The formal aims and objectives are laid out in Section 1.2. A description of the thesis and its contained work is given in Section 1.4. Finally Section 1.5 lists the related publications.

1.1 Motivation

The ever increasing level of product development occurring as companies try to outperform competitors and improve themselves is one of the main reasons behind the need for ever more complex simulations. While complex simulations may take hours if not days to run and be expected to produce beneficial results, it is often the case that only a small budget of evaluations are available for use. These high complexity models can be very expensive to run, both in terms of computational time as well monetary expense. The high computational demand means that it is not always possible to obtain the desired level of output performance. While it is possible to use more powerful computers to overcome some of the computational burden, it is not always a viable option.

A selection of questions arise from this situation including:

- How the design process (engineering workflow) can be improved.
- Whether there is a better design methodology for making more efficient use of the available evaluation budget.
- If it is possible to use the information that is gathered throughout the model calibration stage as a basis for the optimization and make better use of all

available knowledge.
- What is the most efficient way of dividing the budget between the different design stages.
- If it is possible to integrate the model calibration and optimization workflows in order to negate the need for splitting the available evaluations.

The proposed solution to this situation is that of developing a new design method. This new method shall efficiently incorporate both the design steps of model calibration and optimization together. Doing so will allow for greater information sharing and hence reduce the number of function evaluation required while maintaining the overall performance of the system.

It is hoped that this study will have an impact on the scientific community by providing new methods that can be used to incorporate multiple fields of work bringing together ideas from model calibration and optimization. Introduction of new cooperative methods that aim to guide users into having a greater understanding and control over how resources need to be allocated. Additionaly it will help to get people to look at the overall process of design rather than just their own specialist areas.

Providing members of industry with a tool that can aid in reducing the cost of the design process when working with expensive models will prove to be impactfull on how such work is carried out. It aims to add many advantages such as reducing the cost of development and improving the amount of development that can be done with a set budget of evaluations. These issues have been shown to have particular interest for engineering companies such as Jaguar Land Rover who use highly complex simulation during their development processes. Through the incorporation of such newly developed methods companies would have greater flexibility within their design process.

1.2 Aims and objectives

This project aims to develop a method by which the efficiency of the function evaluations that are used within the model calibration and optimization design steps can be improved. The aim of the project can be subdivided down into three objectives:

- Formalise the joint problem of 'model verification, validation and optimization' mathematically
- Develop an abstracted simulation of the problem

- Develop a method to re-use or combine information from historical model builds during optimization

1.3 Contributions

The key original contributions towards the fields of model calibration and optimization which were discovered within this thesis are:

- The proposition of a new mathematical formulation to express the problems of model calibration and optimization as a single unified problem.
- The extension of existing optimization benchmark problems to incorporate calibration parameters to allow for the testing of methods which consider both the stages of model calibration and optimization. Additionally the created multi objective problems were expanded to included cases for both when model error is present as well as when it is absent.
- The creation of a new component, *s_signal*, for the WFG framework that incorporates model parameters. This new component was designed so that it would possess adjustable complexity and could not be easily approximated via a simple surrogate model.
- The proposition of a new alternating methodology for solving the combined problems of model calibration and optimization. It was designed with the aim of obtaining improved knowledge of more relevant area of the output space and hence ensure that the outputs obtained lie as close as possible to the true front.
- Extension of the alternating methodology for use with a small evaluation budget. This was achieved through the incorporation of surrogate models within both the calibration and optimization stages.
- Assessments for the performance of both the new alternating method as well as its surrogate version were performed. In both cases they are compared to a comparable setup in which the classical approach, of performing the two stages in series, was implemented.

1.4 Description of the thesis

This section presents an overview of the structure present within the thesis chapters, first briefly then in more detail. Chapter 2 presents background information

and an overview of the literature relevant to this research. Chapter 3 describes a study performing a component level investigation into the multi-objective evolutionary algorithm based on decomposition (MOEA/D). Chapter 4 looks at forming a mathematical formulation of the combined problem and laying out a real-world problem within it. Chapter 5 goes through aspects related to benchmarking looking at both performance metrics and test problems. Chapter 6 introduces an alternating approach for tackling the combined problem which is further developed in Chapter 7 where a more limited evaluation budget was considered. Chapter 8 concludes the thesis and discusses potential areas of future work.

- Chapter 2 presents a literature review covering the fields of model calibration and optimization. The chapter starts by coving background knowledge that is relevant to implementations from both the fields of calibration and optimization. This is followed by an introduction to model calibration which also looks at performing calibration via statistical inference. Methods that both use likelihood functions, such as Markov chain Monte Carlo (MCMC), are examined along with cases which do not, approximate Bayesian computation. The discussion of calibration is concluded by an assessment of cases in which it is used for computer models. The next field to be looked at within the literature review is that of optimization. Within the optimization field both the single and multi-objective cases are discussed along with the use of surrogate and dynamic methodologies. The review of optimization finishes with a look at expensive multi objective optimization before moving on to examine literature which attempts to consider the combined case. This chapter is finished by presenting the research gaps that are found.
- Chapter 3 describes the examination carried out within a component level investigation of the MOEA/D algorithm. After presenting a thorough introduction to the algorithm along with the components it is comprised of, the implementation that is carried out is given. There were two main areas investigated within the work presented, these being to determine the impact of information sharing within the MOEA/D algorithm and to see how different forms of normalization effect the algorithms performance. After presenting the results of these investigations along with some interesting variants, a discussion of the findings took place and the chapter is finished with some concluding remarks.
- Chapter 4 introduces a new mathematical framework for laying out the combined problem of model calibration and optimization. The framework is presented along with a toy formulation that demonstrates how it can be applied. Once the framework is laid out possible methods by which such a combined

problem could be solved are discussed. The final area looked at within this chapter is the laying out of a real-world example which has been set within the new combined problem framework.

- Chapter 5 describes aspects of the benchmarking process that are necessary later when assessing new methodologies. The use of performance indices within the optimization literature is discussed before a selection of different performance metrics are laid out. After this the chapter moves on to detail a proposed method for altering pre-existing test problems for use with the new case, where they are required to possess both parameters and control inputs. A selection of both single and multi-objective problems are updated using the devised method to form a pool that can be draw from for later testing. The chapter goes on to look at how a new component for the WFG framework was developed, with the aim of providing properties which the current test problems did not possess.
- Chapter 6 presents a new alternating methodology for tackling the combined problem of model calibration and optimization. It begins by setting out the implementation of both methods along with high level schematics. The design choices are laid out along with the performance metrics used and the setup of the test functions. Results for the DTLZ1$_\theta$, ZDT1$_\theta$ and WFG2$_\theta$ functions are presented and discussion.
- Chapter 7 provides an extension of the alternating method in which the use of surrogates is examine with an aim to allowing for greater information sharing. This extended version is aimed at tackling expensive multi objective problems for which only a small evaluation budget is available. The chapter again begins by presenting the new methodologies for the classical and alternating methods, along with relevant implementation information. This is followed by additional information about the algorithm setup before results are presented. The presented results are broken down into four section, with the first analysing an initial investigation into values used to determine part of the setup. After this the results of the alternating methods are presented followed by those achieved by the classical approach. Finally, a comparison of the two methods is presented.
- Chapter 8 concludes the thesis by laying out the key findings and discussion points before presenting future avenues of research.

1.5 Related publications

Parts of the work presented within this thesis are also available within the following publications:

2019

Conference paper

Oliver P. H. Jones, Jeremy E. Oakley, and Robin C. Purshouse. 2019. Toward a unified framework for model calibration and optimization in virtual engineering workflows. In 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC) (pp. 3148-3153). IEEE.

2018

Conference paper

Oliver P. H. Jones, Jeremy E. Oakley, and Robin C. Purshouse. 2018. Component-level study of a decomposition-based multi-objective optimizer on a limited evaluation budget. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18), Hernan Aguirre (Ed.). ACM, New York, NY, USA, 689-696.

Chapter 2

Literature review

The virtual design process stretches over multiple fields of research and areas of study. For this work the review of the literature focuses on two main areas of interest, being, work from the optimization and model calibration communities. Other areas such as model building, decision making and higher-level system design play a part in the overall process but have been deemed too broad for this work. To recap what was mentioned before, we have a specific interest in problems which are computationally expensive and are looking for ways by which the overall virtual engineering design flow could be made more efficient.

Learning more about the methods used to perform calibration and optimization will allow us to identify the current state of the art. This chapter begins by looking at population initialization methods and types of models. These are examined first as they are relevant to both the fields of model calibration and optimization and are used within many methods.

This general background is followed by an overview of model calibration which leads into looking at calibration via statistical inference and concludes by looking at calibration methods for computer models. The third section of this chapter presents an overview of optimization. Within this the different types of optimization ranging from single objective to expensive multi-objective cases are discussed.

The fourth section looks at some of the literature that depicts possible crossovers between the fields of model calibration and optimization. Finally, the last section presents the research gaps which were identified from the literature.

2.1 Background

Before beginning the examination of the literature pertaining to the main areas of interest for this work it is first necessary to examine a few topics which will be present while not necessarily being the main focus. The first of these areas looks at the different methods by which a population of points can be obtained. This is followed by an overview of possible model types that are often used.

2.1.1 Population selection methods

The choice of an appropriate initialization method is an important one and can have a major impact on the performance of an algorithm. The selection of an appropriate population is necessary within both main fields under examination within this work. If considering initialization methods for engineering design the input space to be sampled would be representing either a design variable or environmental uncertainty. Looking at it from the perspective of carrying out

calibration the input spaces to be sampled for initialization would be the model parameters as well as potentially control inputs if none had been previously obtained.

There are many methods to determine a set of initial starting points, most of which fall within one of two groups, either space filling or criterion based. One of the most common methods seen within the literature is a space filling design called Latin Hypercube Sampling (LHS) (Damblin et al., 2013). From examining optimization papers and selecting twenty-eight that used an initial sampling method, it was found that twenty of them used Latin hypercube sampling with another five using random sampling to decide upon their initial set of points. With the rest using alternate methods such as equal spacing and the Halton sequence. There are many approaches for which Latin hypercube sampling has been used, such as the ParEGO algorithm by Knowles (2006) which is discussed later. Latin hypercube sampling can be used when the range of possible input values is known and is capable of being scaled for use with multi-dimensional problems. Another of its advantages is that of ensuring that the whole search area is well covered.

The process of performing Latin hypercube sampling can be broken down into three steps, first, subdivide each dimension of the search space into N regions of equal size. Next select a point at random from within each of these regions. The last step is to randomly combine together points from each of the separate dimensions to produce points that now reside within the multi-dimensional space. A simple two dimensional example of this is shown in 2.1.

A modified version of the Latin hypercube design, presented by Morris and Mitchell (1995), called the 'Maximim Latin hypercube design' aims at acquiring an optimal set of initial points. The basic implementation is similar to that of the Latin hypercube sampling method with the main difference being that a design criterion, maximizing the minium distance between design points, is used to select the best version of the Latin hypercube design after it has been run a set number of times.

Huang et al. (2006) laid out a selection of alternative methods for determining the initial population of points. These include the space filling method of the Monte Carlo design and the design criteria based method of maximum and minimum distance. The Monte Carlo design (Hastings, 1970) works by randomly selecting points within a pre-specified region. When compared to Latin hypercube sampling it has the disadvantage of requiring a relatively large number of points if a reasonable cover of the search space is necessary. This means that a larger number of evaluations is required which when done on a complicated model can lead to a high computational expense.

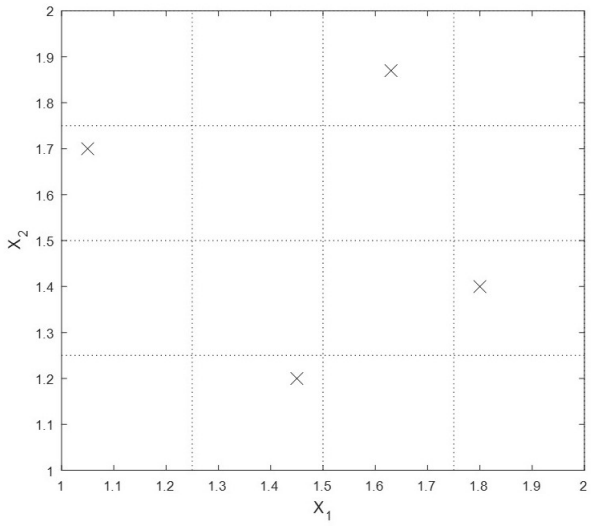


Figure 2.1: An example of Latin hypercube sampling within a two dimensional input space for which four points have been selected. For both dimensions X_1 and X_2, each of the four defined regions only contains a single point which is obtained uniformly at random from within the region.

The central composite design method demonstrates one of the issues some initialization methods face. It decides upon the initial sampling points by taking a centre point and then '2^k' Factorial points and '2 x k' Star points. For example, in a three dimensional problem you would have one centre point, eight factorial points and six star points. This can be seen in Figure 2.2. The issue that the central composition method presents is that of having multiple points corresponding to a single value in each of the input dimensions. Thus, if it is found that one of the dimensions has a negligible effect on the output, all points that have the same values in that dimension are now obsalete, resulting in wasted evaluations. Alternative methods such as the Latin hypercube sampling method and lattice design have gained popularity as they avoid this problem.

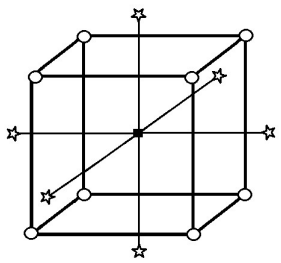


Figure 2.2: Three dimensional Central Composite Design

2.1.2 Model types

When considering models, it is important to distinguish between high fidelity models and surrogates. High fidelity models focus on being accurate to the real-world system they are representing. They are often formed of a complex set of equations and can have a high computational cost to run. Surrogates, also referred to as meta models, are data driven models which are used to represent a more complex system. Unlike high fidelity models which are often custom made for a specific real world problem surrogates are often based of standardised formulations. There are many types of surrogate models present within the literature for both optimization and model calibration. In this section three of these methods are examined, the Polynomial, Radial Basis function and Gaussian process models. These were selected due to their popularity within the literature and the fact they can be applied to a wide variety of problems.

Polynomial model

Polynomial models have been around since the early 19th century and have been used widely in all areas of research. They replicate the expected behaviour of a model using a polynomial equation of length n (see equation 2.1). A polynomial model can then be used to predict output values for new inputs, with determinable confidence. The basic form of a polynomial regression model is as follows:

$$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n + e \tag{2.1}$$

with unknown coefficients a_0 ... a_n which are to be determined and e the error term that represents the difference between the polynomial output and the true output. A common method for fitting a polynomial model to a data set is through the use of the least squares method. This method works by minimizing the sum of squared errors, the difference between true points and model estimates. An example of the polynomial model in use can be seen in the EANA algorithm proposed by Liang et al. (2000). In the case of the EANA algorithm, even though it managed to outperform both the Improve Fast Evolutionary Programming (IFEP) and Evolution Strategies (ES) it was compared against, it still took a fairly large number of evaluations, over 1000, in most cases to run.

Radial basis function

The Radial Basis Function (RBF) is a low-cost function that can be used during surrogate modelling. There are many different types of RBF that can be used when producing a model; Gaussian, linear, multi quadratic and thin plate spline

are just some examples of the options available. A surrogate model is produced by combining multiple RBF's into a RBF network, sometimes known as an artificial neural network (Broomhead and Lowe, 1988). Provided a large enough number of RBF are used, the network will be capable of providing a reasonable approximation of the test function. Gutmann (2001) presents a case in which he uses a RBF for global optimization. His method works by defining a utility function that represents the performance of the RBF model. Using this the optimal point from a set of given inputs can be determined and the new test point selected.

Gaussian processes and kriging models

Gaussian processes (Rasmussen (2004)) are sets of random numbers that have a defined mean and covariance function. A Gaussian process is a stochastic function and can be expressed as:

$$f \sim GP(m, k) \tag{2.2}$$

where m is the mean and k is the covariance of the function. In order to construct a Gaussian process for modelling it is necessary to have a set of known points from which the initial mean and covariance matrix can be found. As more points are obtained through experimentation or evaluations of the true function the model generally improves as the quantity of uncertainty is being reduced. One potential issue that can occur is over training the Gaussian process model. If the process being modelled has a high level of noise or fast oscillations, over training occurs more often and becomes especially problematic. Kriging is a specific case of using Gaussian processes and usually refers to when only a single realization of the process is obtained, often with a mean of zero.

There are numerous variations of the kriging methodology implemented within the literature, three of them are presented by Lebensztajn et al. (2004). The first approach Lebensztajn et al, discuss is the 'Geostatistics approach', which involves estimating a random function and then correcting it through the application of a weighting. In order for this to work, the error needs to be minimised and the expected difference between the estimated random function and the actual random function must be zero. The most important factor to consider in this method is the choice of a suitable covariance function. It is suggested in the paper that a Gaussians or the popular thin elastic plates model (TEPM) as seen in Equation 2.3 should be used for the covariance function.

$$cov(x_1, x_2) = |x_1 - x_2|^2 log|x_1 - x_2| \tag{2.3}$$

The second approach discussed is that of the Maximum Likelihood Estimate (MLE) approach. This approach works by incorporating two functions, the first function $f(x)$ is used to incorporate the normal tendencies of the function to be modelled. The second function $Z(x)$ is a zero mean random process for which a correlation matrix is derived. This is done using a correlation function, often a Gaussian of the form:

$$R(x_i, x_j) = e^{\sum_{k=1}^{N_{par}} -\theta_k(|x_i - x_j|_k)^2}, \tag{2.4}$$

as shown by Lebensztajn et al. (2004), where N_{par} is the dimension of the problem and θ_k is an internal parameter used to represent how the data is correlated in a given direction. This is performed for each combination of x values and the results are stored within the correlation matrix R. Using this correlation matrix the MLE is then obtained and used to calculate the estimated y value, $y^*(x)$, obtained from combining $f(x)$ and $Z(x)$.

$$y^*(x) \;=\; f(x) + Z(x) \tag{2.5}$$

The third approach that Lebensztajn et al proposed is a non-probabilistic kriging methodology called 'Basic Kriging'. In this case it is assumed that the value at each point can be calculated from two functions $w^T h(x)$ and $c^T p(x)$, where $w^T h(x)$ represents any fluctuations around the general tendencies and $c^T p(x)$ represents the general tendencies of the function. Both w and c are unknowns, which must be calculated so that the output at known inputs passes through the observed output points, while attempting to minimise any fluctuations. A comparison of the three kriging methodologies was performed and it was found that the optimal solution was produced by the Maximum likelihood estimation approach. The approach managed to obtain values for the x positions which were nearest the true optimal point.

2.2 Model calibration

This section looks at the first of the two stages of interest from the virtual engineering lifecycle, model calibration. The purpose of model calibration is to determine the values of a set of model parameters such that the difference between the model and the real-world system is minimised. When performing model calibration, the inputs to the model that is undergoing the calibration, are split into two groups, the calibration inputs, θ, and the control inputs, x. The calibration inputs possess a fixed value which can be changed during the calibration process, whereas the

control inputs are chosen to have a known value that relate to acquired observations and include inputs that could change depending upon the calibrated model. The computer model can be defined as,

$$f(x, \theta), \tag{2.6}$$

with physical observations,

$$\{y_i, x_i\} \quad for \quad i = 1, \ldots, n \tag{2.7}$$

where the goal is to find θ such that,

$$f(x_i, \theta) \approx y_i \quad for \quad i = 1, \ldots, n \tag{2.8}$$

Both the inputs and outputs can be either a single value or a vector of values. It is important to note that even when a model has been built there are many reasons why the validity of the model may be threatened. Such reasons can include:

- Temporal variation - when the data is old and changes have happened since obtaining them,
- Spatial variation - when values differ depending on where they were obtained,
- Heterogeneity - when data for a subgroup of the population is not available.

The methods used within calibration are generally broken down into two classifications, either being frequentist or Bayesian. During this section, the process of performing calibration via statistical inference is first discussed. This is followed by a look at methods that both use likelihood functions, MCMC, and an example method that does not, Approximate Bayesian computation (ABC). Finally, an overview of calibration methods used for computer models is presented.

2.2.1 Calibration via statistical inference

In order to be able to perform calibration via statistical inference it is first necessary to assume that the physical observations can be related to the model by,

$$y_i = f(x_i, \theta) + \varepsilon_i \tag{2.9}$$

where ε is an error with some probability distribution, either the same or differing, that is assumed for $\varepsilon_1, \ldots, \varepsilon_n$. Once the problem has been laid out in this form it is possible to use 'standard' statistical inference methods. The two presented here are the Maximum likelihood method followed by Bayesian inference.

Maximum likelihood inference

The Maximum Likelihood Estimation (MLE) is one of the most used methods for performing calibration when a probability model is available for the parameters (Rossi, 2018). The MLE is based on the maximum liklihood principle which states:

"Given a random sample $X_1, \ldots, X_n$ and a parametric model $f(x_1, \ldots, x_n; \theta)$, choose as the estimator of θ, say $\hat{\theta}(\mathbf{X})$, the value of $\theta \in \Theta$ that maximizes the likelihood function."

Using this principle, it is possible to define the MLE of a parameter θ as,

$$\max_{\theta \in \Theta} L(\theta) \tag{2.10}$$

where $L()$ is a likelihood function. There is a vast quantity of literature covering the MLE ranging from high level overviews, such as Myung (2003), to much more specific case studies, for example Excoffier and Slatkin (1995) work on Molecular Haplotype Frequencies in a Diploid Population. A comparison between maximum likelihood inference and Bayesian inference which is discussed in the next section can be seen in the paper by Beerli (2006).

Bayesian inference

Bayesian inference has existed for a long time with the concepts used originating from the Bayes (1763) paper which was edited by Richard Price before being posthumously presented. Laplace continued the development and use of Bayesian statistics within fields including astronomy, meteorology and population statistics. After Laplace, its use diminished until Jeffreys rediscovered it in the 1930's. The field of study was not well comprehended until around 1960 at which point it started to grow more widespread during the 19th century. A more detailed history of Bayesian statistics can be seen in Jaynes (1986) paper.

Bayes' theorem comes from a combination of two of the rules of probability. Assuming that there are two conditional random variables A and B, by combining the product rule of probability,

$$p(A, B) = p(B|A)p(A), \tag{2.11}$$

with the symmetry property,

$$p(A, B) = p(B, A), \tag{2.12}$$

the equation known as Bayes' theorem,

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}, \tag{2.13}$$

can be obtained. In these equations,

- $p(B|A)$ is the posterior probability, this is the probability after the current evidence is obtained.
- $p(A|B)$ is the likelihood, which is the probability of observing the evidence generated by a model using a set value of A.
- $p(B)$ is the prior probability, which is the estimate of the probability before the current evidence is obtained.
- $p(A)$ is the marginal probability, this is the probability of observing A.

Bayes' theorem provides several advantages such as providing a natural way of combining prior information with data and providing a convenient setting for a wide range of models. It also can naturally incorporate uncertainty into the modelling process. Some of the disadvantages of the Bayesian approach is that it does not tell how to select a prior and it can come with a high computational cost. Another disadvantage is that the posterior distribution can be highly influenced by the prior, which can be an issue when people do not trust the selected prior.

2.2.2 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) sampling methods (Goodman and Weare (2010)) are methods which approximate the posterior distribution of a variable of interest by randomly sampling in a probabilistic space. They are often used with Bayesian statistics because of this allowing for the identification of more complex posterior distributions which are either hard or impossible to solve for analytically. It is important to note that Markov chains are memoryless meaning that all the information needed to predict the next step in the chain should be available in the current step. This means that knowing earlier data will not lead to new information being obtained.

Metropolis Hastings

One of the best know variations of the MCMC method is the Metropolis Hastings algorithm which obtains a sequence of random sample from a probability distribution based on a set of rules. The algorithm was originally proposed in a paper by Metropolis et al. (2004) and then further developed by Hastings (1970) which is how it gained its name. A special case of the Metropolis Hastings algorithm is

the Metropolis algorithm in which the proposal function is symmetric. The basic steps which are present for this algorithm are:

- Initialize the starting point and the probability density function used for finding the next point
- Generate a candidate point using the selected probability density function set around the current point.
- Calculate the acceptance ratio
- Determine whether the new point will be accepted dependent on if the acceptance ratio is larger than a uniformly selected random number between 0 and 1
- Either return to step 2 or conclude if all evaluation budget has been used up.

Once the MCMC algorithm has used up its available budget it is usual to discard an initial proportion of the points (known as the 'burn in' period) to help ensure that those which remain lie in the region of interest. The algorithm allows for some leeway in the selection of points with them not requiring to be strictly better than the previously found one. This means that while the algorithm will tend to stay in high density regions it allows for the movement to lower density regions. This is beneficial as in cases where a local minima is determined it increases the chance for the algorithm to break out making it more likely to find the globally most probable region.

Gibbs sampler

In the case that the distribution being examined is multivariate while it is still possible to use the Metropolis Hastings algorithm there are other methods such as the Gibbs sampler (Liu, 2001) which are more favoured. The reason for this is that when the Metropolis Hastings algorithm selects a new multi-dimensional point, as the different dimensions can have varying impacts, selecting an appropriate jump distribution can be hard. The Gibbs sampler overcomes this by varying each of the components of the point separately. Through doing this a separate sampling distribution can be assigned to each of the components of the point. When using the Gibbs sampler each of the components is updated before moving on to the next step and repeating the process.

MCMC methods like this possess a couple of disadvantages including the fact that the samples are correlated. If independent samples were desired, then as the results are they do not necessarily correctly represent the distribution. This can be mitigated to some extent by only using every n^{th} sample, however in doing so a large proportion of the points which were acquired are discarded. On the over

hand MCMC methods are better suited to handling high dimensional spaces for which other methods tend to find the rejection rate increasing exponentially.

2.2.3 Approximate Bayesian computation

Approximate Bayesian computation (ABC) is a set of methods based in Bayesian statistics which do not require the use of a likelihood function (Sunnåker et al. (2013)). This proves an advantage when working with large complex models for which a likelihood model can be hard to derive and expensive to run. An overview of the maths used within one of the Approximate Bayesian computation methods, ABC rejection, is now detailed.

In order to carry out ABC rejection algorithm it is necessary to obtain a set of sampled parameter points θ from the prior distribution. A dataset $\widehat{D}$ can then be obtained from simulation using the statistical model M at a chosen value of θ.

$$\widehat{D} = M(\theta) \tag{2.14}$$

The performance of the model is then determined by looking at the difference between the simulated data $\widehat{D}$ and the observed data D. When making a decision a tolerance is used as, it is unlikely except with the most simplistic of models that the simulated and observed data will coincide exactly. The decision of whether or not to accept the parameter values θ is therefore decided by the condition:

$$\begin{gathered} \rho(\widehat{D}, D) <= \epsilon \\ \text{Where } \epsilon >= 0 \end{gathered} \tag{2.15}$$

The probability of this condition holding is often greatly decreased by an increase in dimensionality. Due to this, a method of using a set of summary statistics $S(D)$ which comprises of a lower dimensional set of data was developed. This set of summary statistics has been designed to capture as much of the data present within D as possible. From using the summary statistics the condition for acceptance changes to:

$$\rho(S(\widehat{D}), S(D)) <= \epsilon \tag{2.16}$$

The final step to performing the ABC rejection algorithm is carried out after a decision of which simulations, and corresponding values from the prior distribution, are to be accepted. It consists of approximating the posterior distribution of θ using the θ values from the accepted simulations.

There are potentially many issues present with the ABC methodologies includ-

ing the previously mentioned 'curse of dimensionality'. While this can cause large problems ranging from having low acceptance rates to a risk of overfitting the data it is possible to combat using methods of model reduction or by using a method to speed up the rate at which parameters are looked at. In relation to this, the use of summary statistics can cause issues when they produce too large a difference from the original data rendering the results meaningless. There are other issues present such as the possibility of conclusions being sensitive to the choice of the parameter range and priors, as well as the use of non-zero tolerance, which could cause a bias in the calculated posterior distribution.

2.2.4 Calibration for computer models

Within this work we are specifically interested in the virtual engineering workflow for which the model being produced is a computer simulation. Due to this being an area of key interest is methods which specifically address the fact that the function $f()$ is a computer model. There are two main features that will require consideration when examining potential methodologies, these being,

- The function may be computationally expensive
- There may be a discrepancy between the model and physical system

A state of the art method for tackling such problems was presented in a paper by Kennedy and O'Hagan (2001). This paper laid out a Bayesian approach for tackling the calibration of computer models. They aimed to improve upon that of the traditional method through designing a process that corrects itself while also incorporating all sources of uncertainty. Within their work they used a Gaussian process model which possesses the advantage of being cheap which allows the function to be treated as known and for the uncertainty to be ignored. Through doing this the issue of problems being computationally expensive is countered.

Looking at alternative methods Wong et al. (2015) presents a frequentist approach for performing computer model calibration. The proposed framework uses a general semi-parametric data model along side an emulator for computationally expensive systems. It also incorporates the discrepancy present between the physical and simulated system using a discrepancy function. It was determined that the proposed approach should work well provided the model discrepancy was negligible and the model parameters represented physical parameters. The authors acknowledged that this methodology is not suitable for all problems but could be useful in cases where there is no good prior knowledge available.

2.3 Optimization

The second stage of the design process that is of interest to this work is that of optimization. Optimization is the process of determining the best solution, defined by some form of criteria, from a set of possible solutions. In the case of this work it refers to obtaining the best possible output from a model through the selection of model inputs. Optimization can be applied to a wide range of problems ranging from simple Single Input Single Output (SISO) models to Multi Input Multi Output (MIMO) models. The general mathematical formulation of the optimization process for SISO is defined as,

$$\begin{aligned} \text{Minimize } z &= f(x, \psi) \\ \text{s.t. } g(x, \psi) &\leq 0 \\ h(x, \psi) &= 0 \end{aligned} \tag{2.17}$$

$$\text{Where } x_{min} \leq x \leq x_{max}$$

In the formulation $z = f(x, \psi)$ is the objective function, $g(x, \psi)$ is the inequality constraint and $h(x, \psi)$ is the equality constraint. The inputs to the model are represented by x, with uncertainty represented by ψ and the output of the function represented by z. The formulation can be expanded to allow for multiple inputs or outputs by making the variables into vectors. If this was to be done then each of the new variables can be defined as,

$$\mathbf{x} = (x_1,\ x_2,\ x_3,\ ...,\ x_n) \tag{2.18}$$

using x as an example and where $\mathbf{x}$ denotes the vector of x's.

When deciding on an optimization method it is important to consider the features of the problem you are going to solve. Features of interest may include:

- The number of model inputs and outputs
- Whether inputs and outputs are continuous
- The fact that f, g and h are implicit functions
- The time it will take to perform each function evaluation
- Whether the number of function evaluations are limited
- If a simple fast model is available
- Whether the function is rough or smooth
- If it is possible to perform multiple evaluations simultaneously
- Whether the problem is stochastic or deterministic
- If there are limits on the model inputs or if it is unconstrained

From the features above, there are some aspects that will need to be considered for this work. Firstly, having models that have multiple inputs causes issues due to increased dimensionality. The main issue is that when only a limited number of evaluations are used, as the dimensionality increases, the unknown regions between points increases. The fact that f, g and h are implicit functions means that the problem is being treated as a black box. For black box problems it is assumed that it is not possible to write down the problems and hence you do not try to analytically solve them. If the function evaluations take a long time to perform or are limited, it is necessary to select a method that allows for this limitation. Most optimization techniques work better for problems in which the output function is smooth; this is due to a rough function having numerous local minima where the ripples occur, leading the optimizer to fall into local minima more often.

2.3.1 Optimization methods

Optimization methods can be broken down into broad fields which include, Linear optimization, Dynamic programming, Convex optimization, and Metaheuristics. Linear optimization, also referred to as linear programming looks at optimizing problems which posses a linear objective function. Additionally, the constraints of the function being examined must also be linear. Linear optimization has been about for a long time and dates back to before 1827 when Fourier published a method for solving them. Currently linear optimization is used widely throughout industry and many other walks of life ranging from the design of diets to making predictions of economic growth. Dynamic programming was first developed by Bellaan (1954). It is a method that looks at simplifying a problem by breaking it down into simpler sub-problems using a recursive method. For optimization this is done through defining a sequence of value functions.

Convex optimization refers to methods in which both the constraint and objective functions are convex (Boyd and Vandenberghe, 2004). Many problems fall within this class and can be effectively solved even if they possess large quantities of variables and constraints. This ability to solve large problems in addition to the relatively low computation power required are some of the main selling points of convex optimization. As would be expected the largest draw back to complex optimization is that it does require the problem to be convex. Within the field of convex optimization, the gradient decent methods are some of the most popular due to their simplicity and effectiveness (Ruder, 2017).

Metaheuristics optimization strategies are high level strategies which control and alter the workings of heuristic strategies in order to obtain the best possible

solution (Voß, 2001). Some of the most commonly used examples of metaheuristic optimization are Evolutionary Algorithms (EA) (Bäck and Schwefel, 1993). These methods have been central to a lot of work done with optimization and have many variations both for single and multi-objective optimization. Another example of metaheuristics are swarm intelligence based algorithms which include methods such as Ant Colony optimization (ACO) (Cáceres et al., 2014, 2015). In cases where the problem possesses a discrete search space one metaheuristic method that is popular is simulated annealing (Kirkpatrick et al., 1983), which can also be applied to problems with continuous search spaces.

2.3.2 Multi objective optimization

A large proportion of real-world problems have more than one objective that is trying to be obtained. This means that when performing optimization, the method selected needs to consider all the objectives and select values that produce the greatest overall benefit. Simply optimizing one objective at a time is likely to lead to a large portion of the optimum values being overlooked. The methods laid out within the literature can be broken down into three main areas that of Pareto based, Decomposition based and Indicator based optimization.

Pareto-based

The first set of multi objective optimization methods that are looked at are those which use the concept of Pareto optimality. The aim when performing this type of optimization is to obtain a Pareto optimal front which occurs when it is not possible to improve one objective without worsening another for a given set of input values, such points would be said to be in a state of Pareto optimality. Depending whether the problem is being formulated as a minimization or maximisation, it may be necessary to alter the form of each of the output variables. Most methods also require that the values be normalised within some limits, this has the benefit of getting rid of any negative values which could potentially be problematic. One of the many advantages of this Pareto-based methods include allowing a single value to represent the overall performance of the system through methods such as the Hypervolume Indicator. This is advantageous as it allows for easy comparison of optimizer performance.

An example of a Pareto based multi-objective optimizer is the non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002). This is a popular Pareto dominance based MOEA and has been used throughout the literature as a comparison for newly developed methods. The NSGA-II function works by producing an

initial parent population which is sorted based on non-dominance. It uses the new population to produce an offspring population, of the same size, through binary tournament selection, recombination and mutation. The crowded-comparison approach is used as it does not require any user defined parameter for maintain diversity among population members. Both populations are combined to form a new population which is sorted by nondomination. This breaks down points into their separate non dominated sets the best of which is assigned to be the new parent population. When the best non-dominated set is not sufficiently large to be used as the parent population extra points are added from the second best non-dominated set, then third and so on until the population is filled. The process then repeats until a stopping criterion is reached or all evaluations are used. It was shown that for nine of the test problems that Deb et al. (2002) applied NSGA-II to it managed to outperform the other methods of PAES and SPEA which it was being compared to.

Another method for performing multi objective Pareto based optimization is with the Multi-objective Genetic Algorithm (MOGA) (Murata and Ishibuchi, 1995). MOGA is not the first optimizer to try and use Genetic algorithms to solve Multi objective problems. An example of such an algorithm is the VEGA algorithm which was developed by Schaffer (1985) this suffered from an issue of obtaining most values at the extreme solutions on the Pareto front however. MOGA works by creating an initial population and then evaluating the points with randomly selected weightings. A temporary set of Pareto optimal solutions is produced before moving on to select a set of string pares from the population. Child solutions are then produced through crossover and mutation with some of the results being swapped out for points from the stored Pareto optimal solutions. The final stages of the optimizer are to test if a stopping criteria have been met, if so the decision maker is presented with the current best set of Pareto optimal solutions. In order to test the performance of MOGA Murata and Ishibuchi (1995) chose to test it against two other multi objective genetic algorithms. The two algorithms used were the VEGA algorithm and the Niched Pareto GA. It was found that MOGA outperformed both the other algorithms although the performance of the Niched Pareto GA did come close to that of MOGA.

Decomposition-based

Another method of approaching multi-objective optimization problems is through decomposition-based methods. Such methods look at breaking down the multi-objective optimization problem into a group of single objective sub problems with the use of scalarization. In recent years, this class of optimization methods

has been growing more popular again since the introduction of the 'Multiobjective Evolutionary Algorithm Based on Decomposition' MOEA/D methodology (Qingfu Zhang and Hui Li, 2007).

The MOEA/D works much like other evolutionary algorithms. There are two main differences present that distinguish it. The first comes from it using the decomposition. At the start of the algorithm a selection of weighting are chosen which are then treated as separate sub problems. The second difference is the inclusion of neighbourhoods. These are initially created at the start of the algorithm and for each weighting consist of a group of other weightings which are close by. While the algorithm is running the neighbourhoods are used within two phases, the first being reproduction. During this stage two points from those within the neighbourhood are selected for use in creating a new child point. The other time that neighbourhoods are used is during the 'Update of Neighboring Solutions' where each point is compared to the newly discovered one using the current weighting and replacing it if found to be better.

There are multiple ways in which scalarization functions can be applied to optimization problems. In addition to simply applying a single scalarization function such as within the MOEA/D algorithm, sometimes it is advantageous to use multiple. Hughes (2007) provides an example of this in his work on extending the Multiple Single Objective Pareto Sampling (MSOPS) algorithm. Each point has a selection of scalarization functions applied with the resulting value being ranked against the performance of the other points. The rankings obtained from all scalarization methods are looked at when deciding upon which point should be selected.

The choice of scalarization function is important to consider when deciding on what characteristics are required by an optimizer. There are two scalarization functions, which consistently appear within the literature. The Tchebycheff function:

$$f(x) = \max_{j=1}^{k}(\lambda_j f_j(x)), \tag{2.19}$$

and the Weighted sum:

$$f(x) = \sum_{j=1}^{k}(\lambda_j f_j(x)), \tag{2.20}$$

The Tchebycheff function is another name for the l_∞ norm. It is a useful scalarization function as it pulls the points towards their assigned direction allowing the user to ensure that the objective space is more fully explored. An example of the

weighted sum (also known as the l_1 norm) can be seen in the work of Ishibuchi et al. (2006) in which they incorporated the weighted sum scalarization function with the NSGA-II algorithm. Unlike the Tchebycheff function, the weighted sum only provides minor influence on the direction of approach to the Pareto front. This means that there is no guarantee as to how well covered the objective space will be.

An altered form of the Tchebycheff function can be seen in use within the ParEGO algorithm (Knowles, 2006) . This augmented Tchebycheff function has a weighted sum (l_1 norm) multiplied by a small value ρ added onto it so as to avoid weakly dominated solutions being selected. It can be seen in Equation 2.30. The maths and concepts behind the augmented Tchebycheff function are explained in the papers by Steuer and Choo (1983) and Dächert et al. (2012).

Indicator-based

The third approach to performing multi-objective optimization is with Indicator-based methods (Zitzler and Künzli, 2004). Indicator-based methods aim to try and either minimize or maximize the value of an Indicator. This can be incorporated into a multi-objective scheme by ranking each point within the population based on the indicator. Once they are ranked a subset of the best points can be selected and improve through acquiring new points, either by reproduction or some other method. Over time, a set of the optimal points based off the selected indicator can be built up.

Many commonly used indicator-based multi-objective algorithms use hypervolume, also referred to as the S-metric as the indicator (detailed explanation of hypervolume present in section 2.3.2). Two such algorithms are the SMS-EMOA algorithm (Beume et al., 2007) and the HYPE algorithm (Bader and Zitzler, 2011). The SMS-EMOA uses a method of choosing which point to discard that is based on the fast-nondominated-sort used by the NSGA-II algorithm. The main difference between the algorithms is that the point to be discarded for SMS-EMOA is the one from the worst front that will have the smallest possible negative impact on the worst front's hypervolume.

SMS-EMOA is shown through benchmarking to be generally more effective on two and three dimensional problems than that of NSGA-II as well as SPEA2 and the ϵ-MOEA algorithm. In addition to outperforming other algorithms on benchmark test problems, it has been effectively applied to real world case studies. One issue that the SMS-EMOA does start to suffer from is the computational burden of calculating the value of the S-metric. This is not such a problem when looking at two or three dimensional problems but could become prohibitive if higher dimen-

sions were used. The ideas present within this work have been adapted for use within other algorithms such as in the case of SMS-EGO (Ponweiser et al., 2008).

The HYPE algorithm aims to overcome the issues associated with calculating the S-metric by using a Monte Carlo simulation to find an approximation of the hypervolume. The idea that makes doing this feasible is that if using a ranked based approach it is not necessary to find exact values to be able to order solutions. While it would still be better to calculate the values, exactly HYPE is set up so that it will automatically switch to estimating hypervolume when there are more than three objectives. Similarly, to SMS-EMOA, the performance of HYPE is compared to that of NSGA-II and SPEA2 in addition to IBEA. It is expected that these algorithms will not manage to optimize the hypervolume particularly well but were instead selected, as they are a lot faster than most hypervolume-based methods. It was found that the mean performance of HYPE was better than any of the algorithms to which it was compared.

2.3.3 Robust optimization

Robust optimization is a field of study that focuses on minimizing the effects of uncertainty (Beyer and Sendhoff, 2007). There are many situations where it is beneficial to implement this approach, such as when aleatory uncertainty is present within your model. With the presence of parameters that are defined as distributions rather than set values, the implementation of robust optimization provides an opportunity to combat the uncertainty caused by the parameter variation without needing to perform calibration. Robust optimization can be defined in general as an optimization problem where there is a new term $\mathbf{\Omega}$ that represents the uncertainty from all sources. This uncertainty includes input uncertainty, structural uncertainty, and parameter uncertainty. The variations caused by the uncertainty that $\mathbf{\Omega}$ represents means that the objective function will give outputs which vary each time it is run and so during robust optimization the output is replaced by an indicator $I(f)$, as seen in Equation 2.21.

$$\underset{x \in X}{\text{Maximise}}\ I(f(\mathbf{x}, \mathbf{\Omega})). \tag{2.21}$$

Before implementing robust optimization, it is necessary to decide upon an indicator to use. The options for indicators can be broken down into three main groups, which are,

- Worst-case scenario - determine the worst case that can be produced from within a bounded domain (Ehrgott et al., 2014)

$$I_{wc}(\mathbf{x}, \mathbf{\Omega}_s) = \max_{\Omega \in \Omega_s} f(\mathbf{x}, \mathbf{\Omega}) \tag{2.22}$$

- Aggregated Value – a combination of possible values gained from the uncertain values determined by an integral measure of robustness (Mourelatos and Liang, 2006). This method uses the expectancy, variance or a combination of the two as the indicator.

$$\begin{aligned} I_{exp}(\mathbf{x}, \mathbf{\Omega}) &= E[f(\mathbf{x}, \mathbf{\Omega})] \\ I_{var}(\mathbf{x}, \mathbf{\Omega}) &= var[f(\mathbf{x}, \mathbf{\Omega})] \end{aligned} \tag{2.23}$$

  where the bi-objective problem is,

$$\min_{x \in X} \left[I_{exp} \ , \ I_{var} \right] \tag{2.24}$$

- Threshold probability – determine how probable it is for the objective function to be better than a reference threshold (Rambeaux et al., 2000). The indicator determines the confidence level where q is the threshold.

$$I_{con}(f(\mathbf{x}, \mathbf{\Omega}), q) = p(f(\mathbf{x}, \mathbf{\Omega}) \leq q) \tag{2.25}$$

Once one of these indicators has been chosen, the results of the indicator replaces the random objective function which would have been used within the optimization. Depending on the representation of the parameter, there are two methods that can be implemented (Langley, 2000),

- Probabilistic - a method which works with distributions.
- Possibilistic - works based on possible realizations of the parameter, often expressed as "scenarios", either
  - A set of scenarios is used within the indicator (e.g. worst case across all scenarios), or
  - Performance against the objectives under different scenarios are represented by additional objectives and/or constraints.

Through implementing robust optimization the effects of uncertainty are minimized. The use of robust methods can present additional issues for problems with expensive evaluations, since typically multiple evaluations for each choice of control inputs are needed in order to understand the variability in the outputs (e.g. via Monte Carlo methods). However, methods are becoming available that attempt to estimate the variability without the necessity for repeated evaluations (Duro et al., 2019).

2.3.4 Dynamic optimization

Another field of study which was identified as potentially being useful for this work is that of dynamic optimization (Kamien and Schwartz, 2012). The main concept behind dynamic optimization is that the problems being considered change over time. Currently the vast majority of problems which are looked at can be classed as 'static optimization problems'. Even with real world problems where there would naturally be some change, they are often taken to be fixed, as it is assumed that any variations present are minimal and can simply be included through some error term. In such cases it is possible for the decision maker to make a single choice, the outcome of which would be used from then on. Dynamic optimization, in contrast, refers to changing problems in which the decision maker will need to make multiple choices.

An overview of dynamic optimization focusing on evolutionary methodologies was presented in the survey paper by Nguyen et al. (2012). In addition to presenting a review of the methods the paper also lies out both information covering benchmarking problems and performance measures for dynamic problems. New performance measures for single objective problems have been produced while classical methods like hypervolume, maximum spread and inverse generational distance are still used with multi objective problems. In such cases however the values of the performance metrics are recorded at set intervals and aggregated over time. A more recent survey by Mavrovouniotis et al. (2017) presents an overview of the use of swarm intelligence for solving dynamic optimization problems.

A lot of the current reasurch focusing on dynamic problems has looked at problems which possess a pre-set way in which they change as time progresses. More recent work by Fu et al. (2014) is of specific interest as it looks at cases where previous choices effect the dynamics of the system. Such a methodology could potentially help during the virtual engineering design process if multiple iterations are required.

2.3.5 Surrogate modelling

Surrogate modelling, also known as reduced-order modelling and meta-modelling, looks at building low cost, simplistic models of more complicated models in order to determine the best parameters with which to evaluate the true model. This enables the possibility of making the most efficient use of the evaluations as possible. An overview of surrogate modelling is presented by Forrester and Keane (2009) and Vu et al. (2017).

The application of surrogate models can be separated into two approaches, integrated use and non-integrated. Integration refers to how a surrogate model is incorporated into the overall design process. Integrated surrogate models effectively evaluate a large number of potential points in order to determine which points will have the largest beneficial effect. The selected points are then evaluated using the true model of the system and the acquired data is used to improve the surrogate. Non-integrated surrogate models refer to surrogate models that are produced using a set number of function evaluations after which they replace the true model of the system and are no longer updated. The decision to use integrated or non-integrated models depends on many factors including how much the surrogate model is trusted and how long the true model takes to run.

The steps necessary to perform optimization with an integrated surrogate model are as follows (Wynn and Bates, 1999):

1. Acquire an initial population of points
   - Initial input array

$$\mathbf{x}_{initial}^{(i)} = [\, x_1^{(i)} \; x_2^{(i)} \; ... \; x_K^{(i)} \,] \qquad i = 1, 2, ..., n \tag{2.26}$$

     n is the number of initial points, K is the number of input dimensions

   - Initial output array

$$\mathbf{y}_{initial}^{(i)} = [\, y_1^{(i)} \; y_2^{(i)} \; ... \; y_L^{(i)} \,] \qquad i = 1, 2, ..., n \tag{2.27}$$

     n is the number of initial points, L is the number of outputs

2. Set up a surrogate model

$$z_m(x^*) = f(x_1^*, x_2^*, ..., x_K^*) \tag{2.28}$$

3. Determine location of new point to be evaluated
4. Evaluate new point on true function

$$y(x^*) = F(x_1^*, x_2^*, ..., x_K^*) \tag{2.29}$$

5. Check to see if stopping criteria has been met. If stopping criteria has not been met repeat steps 3 and 4.

Information on both the selection of an initial population as well as possible choices for a model which could be used were discussed earlier in Section 2.1.1 and 2.1.2. Now that an overview of how surrogate models work have been given

we look at different optimization methods in which they have been used starting with the Efficient Global Optimization algorithm.

2.3.6 Efficient Global Optimization

The Efficient Global Optimization (EGO) algorithm (Jones et al., 1998) uses a combination of Latin Hypercube Sampling and a version of kriging called the Design and Analysis of Computer Experiments (DACE) model (Sacks et al., 1989). The model is used to determine the location at which the largest Expected Improvement (EI) will occur and select a point to be evaluated. Due to using this expected improvement variable the EGO algorithm looks both at areas close to the know points as well as evaluating points in empty regions of the search space. This has the advantage that it prevents the optimiser from focussing too much on local minimums and ignoring the global minimum.

Southall and O'Donnell (2011) presented a practical application of the EGO algorithm, using it for antenna design. They compared its performance to that of a Genetic Algorithm (GA) as well as that of the Nelder-Mead algorithm. While the Nelder-Mead algorithm did perform well, it was out-performed by the EGO algorithm. Testing the EGO algorithm against both a simple and enhanced Genetic algorithm, they found that the EGO algorithm outperformed both of the GA's and provided a higher degree of accuracy with fewer evaluations. An interesting idea that they suggested was the inclusion of the 'linear decreasing' endgame technique with the EGO algorithm. This was shown to both reduce the deviation from the global minimum as well as reducing the variation over multiple runs compared to that of the GA or Basic EGO algorithm.

One of EGO's drawbacks is that as the search progresses and the number of known points increases, the time to perform each iteration of the algorithm increases as well. While this will not have a significant impact for problems that take hours to days to complete, it is an issue when quickly testing the algorithms performance with a simplistic model. Knowles (2006) present a simple solution involving selecting a subset of the points that would be used instead of the full set. The subset consists of two parts, the first being from the best solutions and the second from a random choice without replacement of the other points.

Many people have looked into assessing and improving the EGO algorithm. One area of research that has been explored is the effectiveness of constraint handling, (Habib et al., 2016b), and what type of sampling criteria is most effective within such constrained problems, (Sasena et al., 2002). Another area of work is that of looking for better methods by which new points can be selected, one example of which is the bootstrapped kriging method proposed by Kleijnen et al.

(2012). Other research looks at a higher level view of how EGO should be carried out. Examples of such are work done on the best way to balance the amount of local and global optimization when EGO is used in parallel (Zhan et al., 2017), and the idea of using the sequential kriging Meta-Models (Huang et al., 2006).

2.3.7 Mixed-integer surrogate optimization

The Mixed-Integer Surrogate Optimization (MISO) framework (Müller, 2016) is another approach that looks at solving computationally expensive optimization problems. MISO, similarly to EGO, uses Latin hypercube sampling for acquiring its initial points. The MISO framework can work with a selection of algorithms including sequential radial basis function (SRBS), Dynamic Coordinate search using Response surface models (DYCORS) and Gutmann's RBF method. Two new MISO algorithms, MISO-CPTV and MISO-CPTV-local were proposed. They comprise of a combination of Coordinate perturbation, Target value and, in the case of MISO-CPTV-local, a local search. Within the algorithm each of the three stages are referred to as the 'c-step', 't-step and 'l-step' respectively. Each of the steps have distinct purposes, the c-step looks at find promising regions throughout the whole search space. The t-step identifies the better value from within the promising regions identified by the c-step. The l-step is implemented when both the c-step and t-step can no longer find better values and performs a local search around the current best solution to try to improve its performance.

Algorithms including the efficient global optimization algorithm and the mixed-integer surrogate optimization framework are limited in that they only work for single objective problems. The next section will look at potential ways of overcoming this.

2.3.8 Expensive multi-objective optimization

Within real world problems there are some which prove to be highly expensive to run. This high expense can be due to many things such as the problem being highly complex or time consuming to evaluate. This is problematic from an optimization standpoint as it means that only a limited number of evaluations can be carried out. While many optimization methods do not look at this problem the are some that do.

The ParEGO algorithm which, was first presented by Knowles (2006), is an example of such an algorithm. While other extensions of the EGO algorithm, such as those presented by Ponweiser et al. (2008) and Habib et al. (2016a) exist for use in multi objective optimization ParEGO was specifically designed for expensive

problems and has become popular within the field. It combines the system outputs $f_1 ... f_n$ to form a single value for the system by using the augmented Tchebycheff function (Equation 2.30). Of the three variables within the equation f is the system output, λ is a weighting and ρ is a small positive value which is usually set to 0.05 (Knowles, 2006).

$$f_\lambda(x) = \max_{j=1}^{k}(\lambda_j.f_j(x)) + \rho \sum_{j=1}^{k} \lambda_j.f_j(x) \tag{2.30}$$

The ParEGO algorithm has been compared against other algorithms both within its original paper as well as in others such as Knowles and Hughes (2005) and Knowles et al. (2009). It has been shown the ParEGO algorithm was capable of outperforming alternatives such as the non-dominated sorting genetic algorithm II (NSGA-II), binary search algorithm (Bin_MSOPS) and Tau-Oriented Multi objective Optimizer (TOMO) in all but a few cases. There are many reasons why this might be the case, one of which is that the ParEGO algorithm was design specifically to be able to cope with limited numbers of function evaluations where as most algorithms are not. From the testing it was concluded that ParEGO could make better use of the discovered points due to issues such as Bin_MSOPS missing out when there is only a low density of points at the Pareto front.

There has been research into many aspects of the ParEGO algorithm such as the use of LCB instead of EI (Horn et al., 2015) the incorporation of decision maker preferences (Hakanen and Knowles, 2017) the choice of scalarising norm (Trautmann et al., 2017) and the use of dual kriging (Davins-Valldaura et al., 2017).

2.4 Combined calibration and optimization

Within the literature there are many cases of taking methods from one field of study to aid in improving those from another. One example of this is the use of multi-objective methods to improve pre-existing calibration methods (Li et al., 2010, Zhang et al., 2008). While crossover such as this does occur and can lead to improvements, in general the two fields of study are still kept separate from each other. In the rare cases when a published work considers both the calibration and optimization stages together, they are usually looking at specific case studies. For example, the paper by Gibbs et al. (2010) presents a study on 'Pumping and Disinfection of a Real Water Supply System' and while both stages are discussed, they are treated as separate entities.

Another example of a paper which considers both the optimization and calibration stages is Villarreal-Marroquín et al. (2017) which looks at work on injection

moulding. Examining the paper, it can be seen that while a model is constructed and calibrated, the calibration is not revisited after the optimization has started. Similarly, the formulation of the problem is done as one following the other rather than considering the overall process. While the problem being examined is multi-objective the techniques being used for optimization are simplistic with a grid search being applied with a second iteration in which the search region is refined.

One paper that appears to more deeply examine the problem is a study on common rail diesel Engine calibration and optimization by Qiang et al. (2004). In this paper the authors look at using a combined method of neural networks and Adaptive Network-based Fussy Inference System (ANFIS) to create a combined model. They found that this proved to be a valid modelling method for their case study. The study did not however lay out the problem mathematically or consider other methods that could be effective.

More generally we are also interested in cases where the methodologies used within both calibration and optimization either have the same structure or some linking element. The most obvious of these are cases in which surrogate models are used as they should allow for direct information sharing. Within the paper by Kennedy and O'Hagan (2001) on the calibration of computer models a Gaussian process model was used as a surrogate. This could be potentially used as a link to an optimization method such as ParEGO (Knowles, 2006). Another linking factor between these two methods that can be considered is the method used for initialization. In both cases a form of Latin hypercube sampling was used. If this initial population were shared it could cut down on the number of points used for initialization opening them up for more beneficial uses.

Another example of Model calibration using similar processes to those in Optimization can be seen in Kajero et al. (2016) paper 'Kriging Meta-Model Assisted Calibration of Computational Fluid Dynamics Models'. In this paper, they have proposed a method that is similar to that of the EGO algorithm. It works by minimising the sum of squared errors between the original model and the kriging meta-model through the adjustment of parameters.

This type of combined optimization and model calibration problem also seems to appear within some newer work present in the field of robotics. Specifically, recent work on continuous self-modelling from papers such as that of Bongard et al. (2006) show similarities. In this work they show how robots can be capable of synthesising new models and optimizing their parameters when unexpected changes occur and potentially generate compensatory behaviour. More recently work by Kwiatkowski and Lipson (2019) shows a robot successfully managing to maintain its self-model and continue its operation through an increased data usage

after it has a component replaced with a deformed part. This type of behaviour links closely with the concept of combining model calibration and optimization which is of interest within this work.

Here we have shown that while there is some literature depicting both the stages of calibration and optimization they are still mostly considered as separate steps. While there are methods that have been used within both fields, they have not been considered for combining into a single unified method. The next section looks at gaps that have been identified as being present within the current literature.

2.5 Research gap

From this examination of the literature there are a selection of gaps which have been identified. The first of these comes from looking at the current combined calibration and optimization literature. The work that has been carried out has predominantly focused on implementations for specific applications, also the methods used are often simplistic. It was also normal that when both stages of the design process were presented no further consideration into improving the overall efficiency was given. From this it can be seen that the development of a unified approach using state-of-the-art methods to solve the combined problem would be a novel area of research. Examining how the process could be changed to improve the overall efficiency of the design process rather than just the individual stages would also be novel.

One area identified within the literature that might be able to be applied here is dynamic optimization. It could be used in situations such as when the model is being developed over a period causing multiple iterations of the calibrated model to produced. More specifically, if a situation where the model is being developed in parallel with the optimization process, such that either discrete realisations or a continuously changing model is available, it provides a framework. This is a potential avenue to explore for combining the two stages into a unified methodology.

While within the literature there are a large variety of performance criteria and test problems present there are currently no benchmarking problems that span the two fields of research. Due to this it will be necessary to develop a new set of test problems which possess all desired characteristics before developing and testing new methods.

Another area of interest is that there is potentially a wide range of methods being overlooked that might be more efficient that the current practice of performing calibration followed by optimization in a sequential fashion. While this practice is

effective for many problems this does not mean that it is necessarily suitable for problems in which there are multiple iterations of the model being produced.

Chapter 3

MOEA/D study

3.1 Introduction

This chapter focus on the behaviour of a representative algorithm from one of the main classes of Multi-objective evolutionary algorithms (MOEA) – specifically, we consider the *MOEA/D* algorithm (Qingfu Zhang and Hui Li, 2007) from the decomposition-based family of optimizers. A full version of this work was published at GECCO in 2018 under the title 'Component-level study of a decomposition-based multi-objective optimizer on a limited evaluation budget'. Rather than study the algorithm as a monolithic entity, we adopt a *component-based approach* that allows the impact of different aspects of the algorithm to be analysed in detail (Bezerra et al., 2015, Laumanns et al., 2001, Purshouse and Fleming, 2002). The focus is on how typical component choices that would need to be made when configuring an optimizer for a real-world applications (RWA) affect optimizer behaviour over a small evaluation budget. In this way, we seek to contribute the first known analysis of decomposition components for small budgets, in isolation from the complexity of surrogate-based components – with a view to making some preliminary recommendations for how such a targeted decomposition-based algorithm should be configured.

MOEAs have come to be used widely throughout both the scientific and engineering communities, and are typically classified in terms of the primary selection method used in the algorithm: *Pareto*-based, *decomposition*-based and *indicator*-based (Giagkiozis et al., 2015). Most of the methods that have been developed within each class typically assume that a large budget will exist for evaluating candidate solutions as the optimization process progresses. In many optimizer benchmarking studies, a budget of tens or hundreds of thousands of evaluations is used – for example, the CEC'09 MOEA competition permitted 300,000 evaluations (Zhang et al., 2009).

However, solution evaluation can be an expensive procedure for many RWAs, typically arising from the use of high-fidelity simulations or physical experiments. In the case of high-fidelity simulations, even if the computational costs of running the models can be overcome (e.g. by careful exploitation of high performance computing facilities) then other resource constraints often still remain (e.g. availability of software licenses). In this setting, it is inappropriate to assume that a budget of many thousands of evaluations will be available to the optimizer.

Faced with this issue, algorithm designers have sought to couple surrogate modelling techniques to the optimization process (Jin, 2011). In a loose coupling, the surrogate model is estimated either before the optimization begins, or at scheduled points during the optimization process. In more tightly coupled schemes,

the surrogate model becomes a key component of the optimizer itself. These latter schemes have found particular favour for optimization on very small budgets – typically regarded as between 100 and 500 evaluations (Knowles and Hughes, 2005). The algorithms, such as ParEGO (Knowles, 2006), are typically highly complex in nature, featuring large numbers of configurable parameters. It remains unclear how these parameters should be set for different types of RWA, or how the behaviour of the overall algorithm is related to the behaviour of the underpinning selection method.

The chapter is structured as follows. Section 3.2 gives an overview of the MOEA/D algorithm, including an abstraction of its components. Section 3.3 isolates the different components which are examined in the study and describes the empirical framework used. Section 3.4 presents the results of the experiments which are then discussed in Section 3.5. Section 3.6 draws conclusions and indicates future directions for the research.

3.2 MOEA/D and its components

The *Multi Objective Evolutionary Algorithm based on Decomposition* was first presented by Zhang and Li in 2007 (Qingfu Zhang and Hui Li, 2007). The algorithm, as was discussed in the literature review (Section 2.3.2), is based on the classical multi-objective optimization concept of defining different reference directions in objective-space, and then directing the optimization process along each of these directions. The main innovation in MOEA/D is that information is *shared* between neighbouring reference directions during a single optimization run – rather than performing separate optimization runs for each direction in turn as was used in the classical methods. The underlying hypothesis is that there is some kind of relationship in the neighbourhood that makes the sharing of information useful for the neighbours concerned.

MOEA/D has proved to be a seminal MOEA, with many variants and alternative decomposition-based schemes now in existence. A tightly-coupled surrogate-based addition – MOEA/D-EGO – was proposed in 2010 (Zhang et al., 2010) as an alternative to the earlier ParEGO algorithm which also used decomposition-based principles (Knowles, 2006).

A number of studies have investigated configuration choices for MOEA/D, offering alternatives to the originals proposed in Qingfu Zhang and Hui Li (2007). Parameters examined include choice of norm in the scalarising function (Ishibuchi et al., 2010, 2013), weight vector specification (Qi et al., 2014), and neighbourhood size (Zhao et al., 2012).

3.2.1 Components of MOEA/D

Within the implementation of MOEA/D there are four main stages that can be abstracted as components: (1) initialisation; (2) reproduction; (3) improvement; (4) update. In addition to these aspects, a further area of interest is the normalisation operation used to enable non-commensurate objectives to be compared. How these components fit within the algorithms structure can be seen in the pseudo-code provided in Algorithm 3.1. Of the components present, reproduction, update neighbouring solutions and normalisation are considered as optional and are investigated in Section 3.3. An overview of each of these components is given below along with their corresponding line number form within the pseudo code.

Initialisation. Lines 1-4. The first step that needs to be performed is to initialise the various parameters present within the MOEA/D algorithm. The algorithm begins by defining a set of weight vectors, which are evenly spread throughout the output space. The next step is to set up neighbourhoods which consists of the closest, determined using Euclidean distance, n weight vectors to each weight vector. The final aspect that needs to be implemented within the initialisation is the creation of a set of initial points. Within the MOEA/D algorithm, these points are determined either randomly or through a problem-specific method.

Reproduction. Line 8. After initialisation, the optimizer loops through all of the reference directions. For each direction, the optimizer will select two weight vectors from that direction's neighbourhood and use the points from those two reference directions to perform reproduction. The type of reproduction used by MOEA/D is simulated binary crossover (SBX), during which only one offspring is created. That offspring then has polynomial mutation applied to it.

Improvement/Repair. Lines 10. The improvement stage is used to apply either a problem specific repair or improvement heuristic to the offspring point. This stage is useful as it can ensure that the acquired solution is feasible.

Update neighbouring solutions. Line 13. This section of the algorithm works on ensuring that newly found solutions are used effectively. The new solution is compared to all other solutions related to it through its neighbourhood. The comparison of whether the newly found solution is superior to existing solutions in the neighbourhood is performed by applying the Tchebycheff scalarisation function associated to each of its neighbour's reference directions.

Normalisation. Line 12. While normalisation is not a separate stage specifically stated within the MOEA/D algorithm, it is still carried out and has an effect on the algorithm's performance. In the basic description of the algorithm in the original MOEA/D paper (Qingfu Zhang and Hui Li, 2007) there was no normalisation undertaken prior to scalarisation. However, normalisation was discussed

Algorithm 3.1 Component-level abstraction of MOEA/D

```
1: for 1 : Number of replications do
2:     Initialise parameters
3:     Initialise neighbourhood
4:     Initialise points using Latin hypercube sampling
5:     for 1 : Number of iterations do
6:         for 1 : Number of directions do
7:             for 1 : Number of offspring do
8:                 Optional component: Reproduction
9:                 Mutation
10:                Improvement/Repair
11:                Evaluation
12:                Optional component: Normalisation
13:                Selection
14:                Optional component: Update neighbouring solutions
15:    Calculate and store performance metrics
```

in a separate analysis within the original paper to understand if improved performance could be achieved when working with disparately-scaled objectives. The normalisation is undertaken with respect to the ideal and nadir points. These points are estimated progressively from the solutions that have be found so far: the best values achieved for each objective are taken as the current estimate of the ideal; the worst values achieved within the current best set of solutions across all reference directions are taken as the current estimate of the nadir.

3.2.2 Implementation of components

The baseline MOEA/D algorithm within this chapter has been setup slightly differently to how it was described within Zhang and Li's original paper. These variations are caused by the desire to test different aspects of the optimizer. The chosen initialisation method implemented throughout the chapter is the widely used space-filling Latin hypercube sampling method. While this method does require the user to have knowledge of the limits of the input space, it ensures that the whole of the input space is well covered. In the baseline setup it is also assumed that the ideal and nadir points required for performing fixed normalisation are known to be the correct values. The improvement step simply checks whether decision variables have moved outside of their bounds and, if so, sets them to the closest feasible boundary. It was assumed that components such as the neighbourhood, reproduction and updating are not implemented unless they are the aspect specifically under test. A representative trajectory plot for the baseline algorithm solving the test problem, shown in Section 3.3.2, is presented in Figure 3.1.

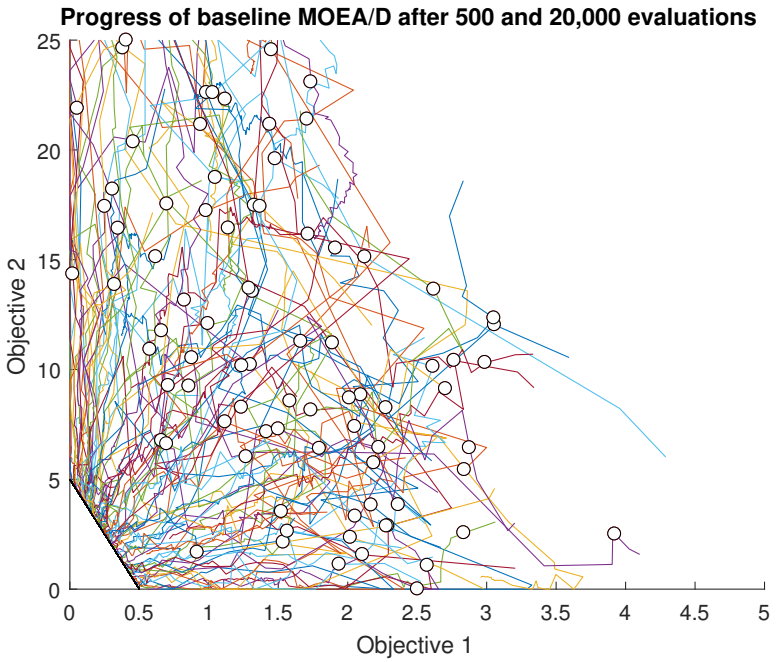


Figure 3.1: Trajectory plot for the baseline algorithm using 100 reference directions and a budget of 20,000 evaluations (progress after 500 evaluations is shown by the circled points). Good convergence is observed for the larger budget.

3.3 Component Investigation

3.3.1 Areas of interest

While MOEA/D has many areas of interest that could be investigated, we focus on two key component choices for RWAs: the first of these is how the inclusion of information sharing through neighbourhoods affects the optimizer's performance; the second area of interest is how different forms of normalization cause changes in the optimizer's behaviour.

Baseline optimizer

The baseline configuration, also referred to as independent, consists solely of a mutation component, with no reproduction, or update of neighbouring solutions. Due to the budget limitations, a very modest number of reference directions has been chosen: 5 in total. The optimizer is run for 100 iterations, within each of which one evaluation is used for each of the 5 reference directions, leading to the total budget of 500 evaluations being exhausted. An elitist $(\mu + \lambda)$ strategy is used for selection across all variants of the algorithm. A $(1 + 1)$ strategy is used in the baseline algorithm. Scalarisation is performed using the Tchbycheff norm.

Impact of sharing information

Three component configurations consider the impact of sharing information between the different subproblems present within MOEA/D - all require the definition of a neighbourhood. The neighbourhood is defined as the adjacent two reference directions. The first approach implements the update of neighbouring solutions component; the second implements SBX reproduction, with three solutions considered (the two parents are chosen at random from both the neighbourhood and the reference direction). The SBX distribution index is set to 20 as in the original MOEA/D paper (Qingfu Zhang and Hui Li, 2007). The third setup, refered to later as mixed, consists of using the neighbourhood for both SBX reproduction and update of neighbouring solutions. This final configuration provide the greatest sharing of information between the different references directions. As with the previous setups, a neighbourhood size of two is implemented in order to maintain consistency.

Impact of normalisation

One of the issues when performing decomposition-based optimization on RWAs is that the location of the ideal point and nadir point, required for normalisation, are generally unknown. In many studies, an assumption is made about the points used for normalisation, such as is done in this paper during the assessment of sharing of information. The choice of using fixed known points for the ideal and nadir point is reasonable as it allows the impact of these neighbourhood features to be examined without having to simultaneously consider the impacts of using estimates of the ideal/nadir points. However, in order to test the impact of normalisation on the optimizer, different normalisation methods are applied to the most effective information sharing setup examined above.

Four alternative normalisation approaches are considered. The first uses a fixed normalisation with limits that are known to work well. The second variation is to not apply normalisation at all (Ishibuchi et al., 2017). The use of adaptive normalisation is the third setup, in which new ideal and nadir estimates are obtained for each iteration of the optimizer. This is implemented in the same way as in the original MOEA/D paper. The final setup looks at whether using a portion of the evaluation budget to determine an approximate value of the ideal and nadir points could be beneficial. In order to determine these two points, a lexicographic optimization methodology is used with a $(1 + 1)$ strategy. This first determines the minimum for one of the objectives, before minimising the second objective while maintaining the value found for the first.

3.3.2 Performance analysis

Test function

A modified version of the DTLZ1 function, initially presented in Deb et al. (2005), has been chosen. It has been altered in order to make the objectives disparately scaled so that some form of normalisation might prove beneficial. The altered DTLZ1 function, referred to here as 'DTLZ1alt', is defined as:

$$\begin{aligned}
&\text{Minimize } f_1 = \frac{1}{2}x_1(1+g)\\
&\text{Minimize } f_2 = 5(1-x_1)(1+g)\\
&\qquad g = \Big[5 + \sum_{i\in\{2,\dots,6\}} (x_i - 0.5)^2 - cos(2\pi(x_i - 0.5))\Big]\\
&\quad \text{Where } x_i \in [0,1], i \in \{1,\dots,n\}, n = 6.
\end{aligned} \tag{3.1}$$

The test problem possesses a linear Pareto front, which stretches from the point $[0,5]$ to $[0.5,0]$. Points on the front can be generated by setting $x_{2,\dots,6} = 0.5$ and taking any choice for x_1.

Performance indicator

The performance indicator used throughout the analysis is the inverted generational distance (IGD), which measures the quality of a non-dominated set in terms of both convergence and diversity simultaneously (Sierra and Coello, 2004). IGD calculates the average Euclidean distance between a set of evenly spaced points which lie along the Pareto front and the closest points to them from the current non-dominated set. A more detailed explanation of IGD can be seen in Section 5.2.3 in which a description and mathematical formulation are presented.

In the analysis, the points along the Pareto front used for IGD are the points where the reference directions cross it when ideal normalization is applied. In order to gain robust insight into how the optimizer will perform, each setup is run 31 times. Box plots are used to indicate the variation in performance, as well as median levels of attainment. In addition to producing a box plot of the IGD obtained after 500 evaluations, a box plot of the *integrated IGD* is also considered, which considers the sum of the IGD obtained over the iterations of an optimizer run. The integrated measure provides some insight into how rapidly convergence is obtained, which is useful context if 500 is regarded as an upper limit on the number of permissible evaluations.

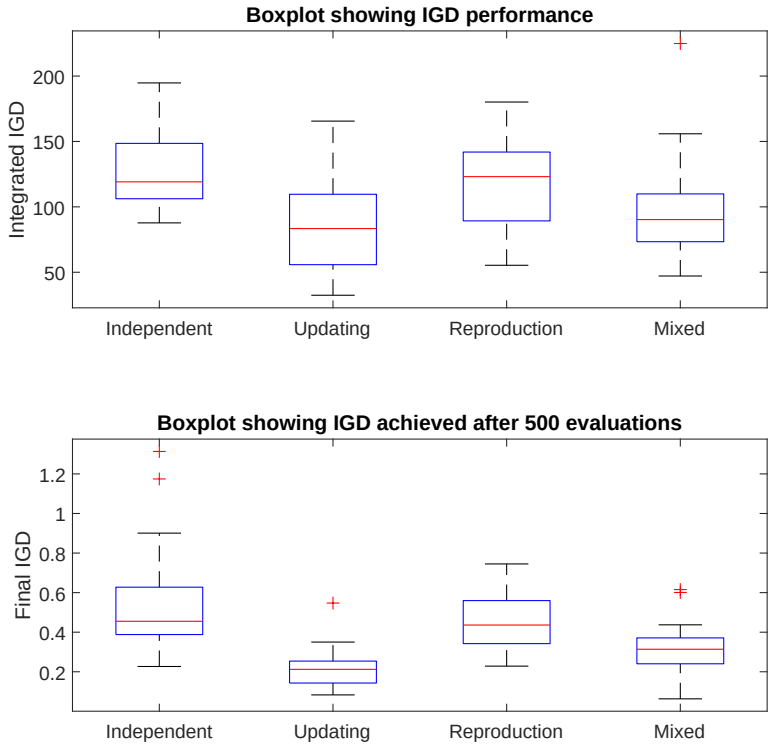


Figure 3.2: Boxplot showing how the use of neighbourhoods to share information between subproblems within the optimizer impacts the IGD. It is evident that the inclusion of neighbourhoods for updating of neighbouring solutions has a positive impact on performance.

Subproblem convergence trajectories

In addition to the IGD metric, we also show the dynamic convergence in objective-space of the optimizer along each of the five reference directions. The trajectories are taken from the run of the optimizer associated with the median IGD result. These trajectories provide useful insight into the dynamic behaviour of the optimizer for different choices of information sharing or normalisation.

Significance testing

The statistical significance of the results (at the 0.05 level) is determined using pairwise Wilcoxon rank-sum tests with Bonferroni correction. The test statistic used is the mean IGD performance for each algorithm after 500 evaluations.

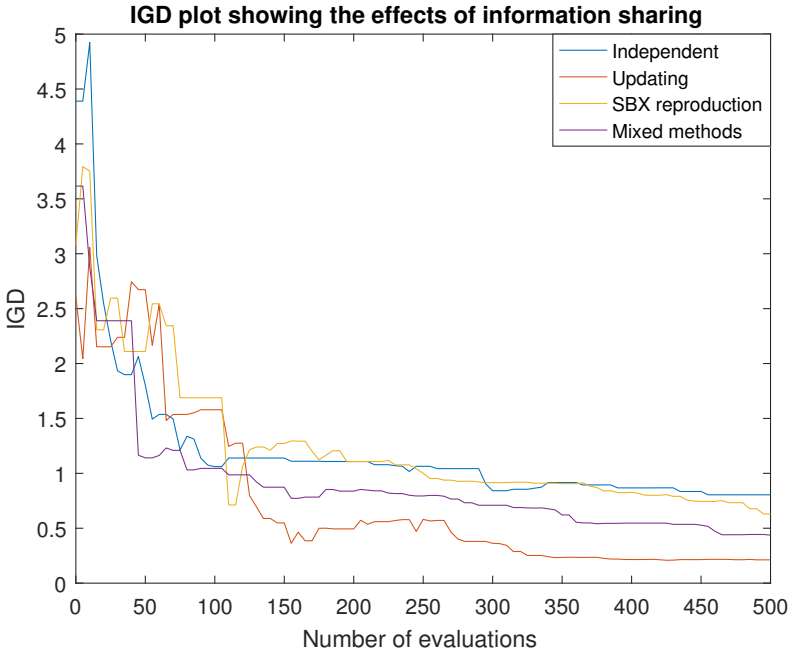


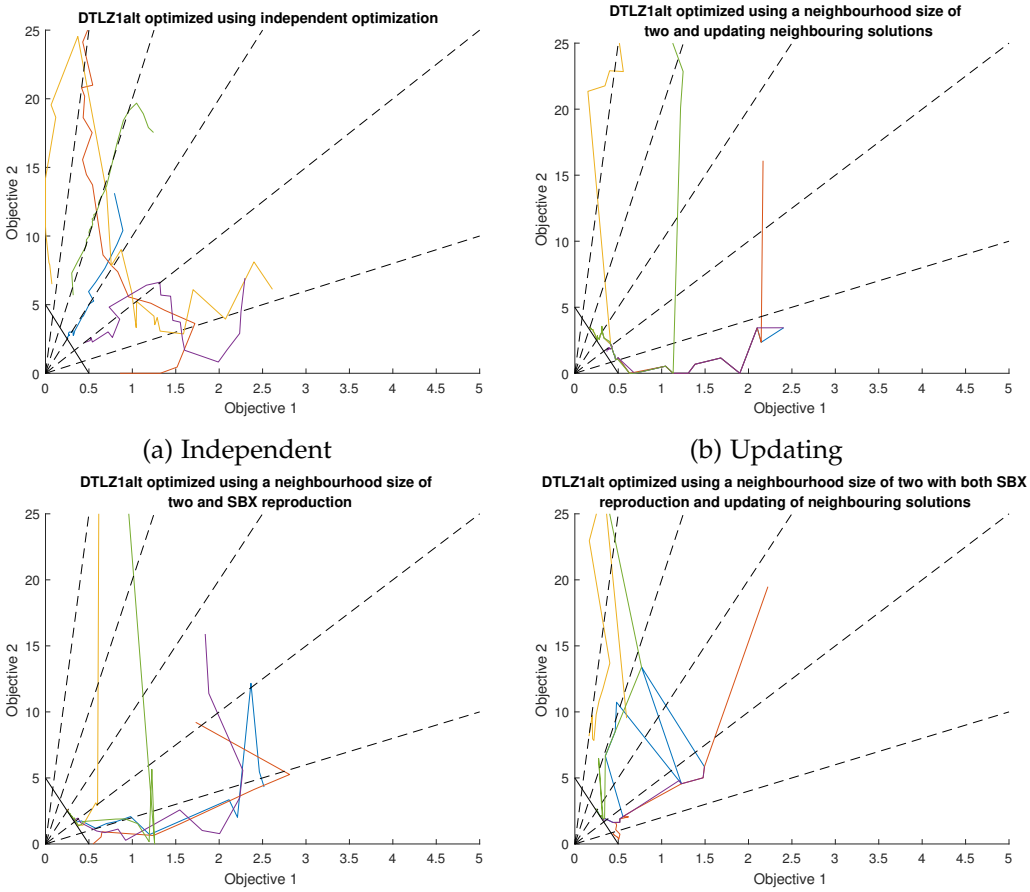
Figure 3.3: Plot showing how the median IGD for different setups progresses over iterations of the optimizer. The positive impact of updating is clearly visible for the majority of the evaluation budget.

3.4 Results

3.4.1 Impact of sharing information

The results of implementing the different methods for information sharing can be observed in Figure 3.2. All relative rankings of algorithms implied by the lower boxplots are statistically significant, except between the independent and reproduction configurations, where no difference in mean performance could be confirmed.

When the subproblems are performed independently, the performance of the optimizer appears to be worse with its final IGD achieving a median value of about 0.45. When the updating of neighbouring solutions component is included, it is evident that the optimizer's performance is positively impacted – with the optimizer achieving a median final IGD of about 0.21. Another point of interest with this setup is that the integrated IGD implies that it was also faster to reduce its IGD, and so could prove to be a good option if there were the possibility that the optimizer would need to stop early. The use of reproduction without the update of neighbouring solutions does not improve the optimizer's performance. While it produces an equivalent median final IGD when no information sharing was



(a) Independent (b) Updating

(c) Reproduction (d) Mixed

Figure 3.4: Trajectory plots of the median runs, based on the integrated IGD. The inclusion of neighbourhoods causes clustering and rapid initial movement.

used, it possesses a worse integrated IGD. This can be more clearly seen in Figure 3.3 which shows the median runs based on the integrated IGD. The final setup examined is the one using a mix of SBX and updating neighbouring solutions. While this setup does perform better than that of either using the independent setup or SBX reproduction, it does not outperform the setup using just the update of neighbouring solutions.

The set of plots in Figure 3.4 show the trajectories of how the subproblems progress within the objective-space. The independent setup in which there is no passing of knowledge seen in Figure 3.4(a) performs as expected when run using the Tchebycheff scalarisation function, with each of the trajectories moving over to their respective reference direction before progressing down it towards the Pareto front. The large jumps induced by updating neighbouring solutions are evident within Figure 3.4(b). It can be observed that when points lie far from their reference directions or far from the front they quickly jump to a closer point causing rapid movement towards the front. Once near the front, the trajectories diverge with each moving over to its relevant reference direction. What appears to be a similar behaviour can be seen in Figure 3.4(c) which shows the median run of the effects of using SBX. The grouping caused by SBX reproduction appears to be an issue when the points draw close to the Pareto front, as they remain clustered together. Finally, Figure 3.4(d) shows the effects of using a mixture of reproduction and updating the neighbouring solutions. Similar to before, the points progress down towards the Pareto front while being gathered together; once they reach the front they try and spread across it, but only have limited success.

3.4.2 Impact of normalisation

The results of testing the impact of applying different normalisation methods are presented within the boxplots in Figure 3.5. As previously, all relative rankings of algorithms implied by the lower boxplots are statistically significant, except for the comparison between the adaptive approach and no normalisation, where no difference in mean performance could be confirmed. The fixed normalisation that has been used throughout the chapter so far shows good performance in both the boxplot of the integrated IGD as well as that of the final IGD. As well as achieving a very low IGD value, the results for fixed normalisation have only a small amount of variation. Using no normalisation, the second setup has a worse performance with both the integrated and final IGD obtained being much higher than that of fixed normalization. A point of interest is that when no normalisation is used, there is also little variation in performance across runs of the optimizer. When adaptive normalisation is used, the median integrated IGD and final IGD

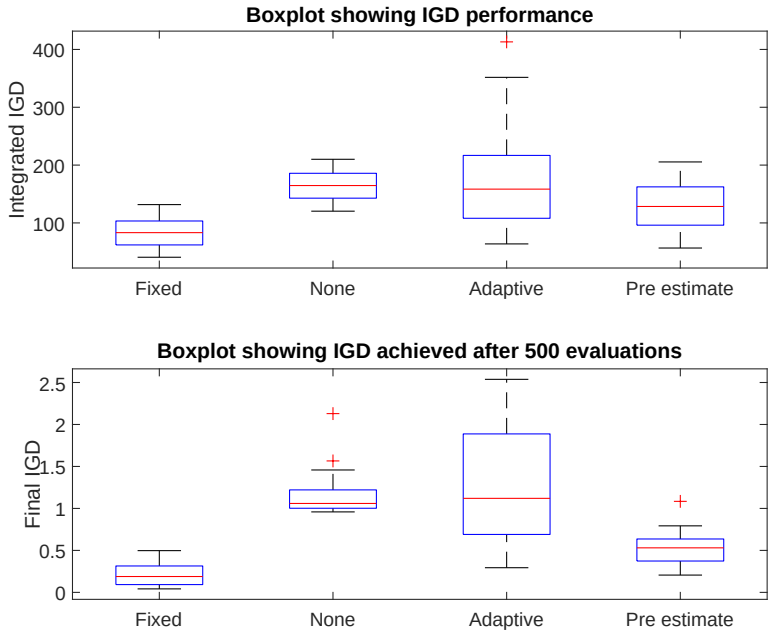
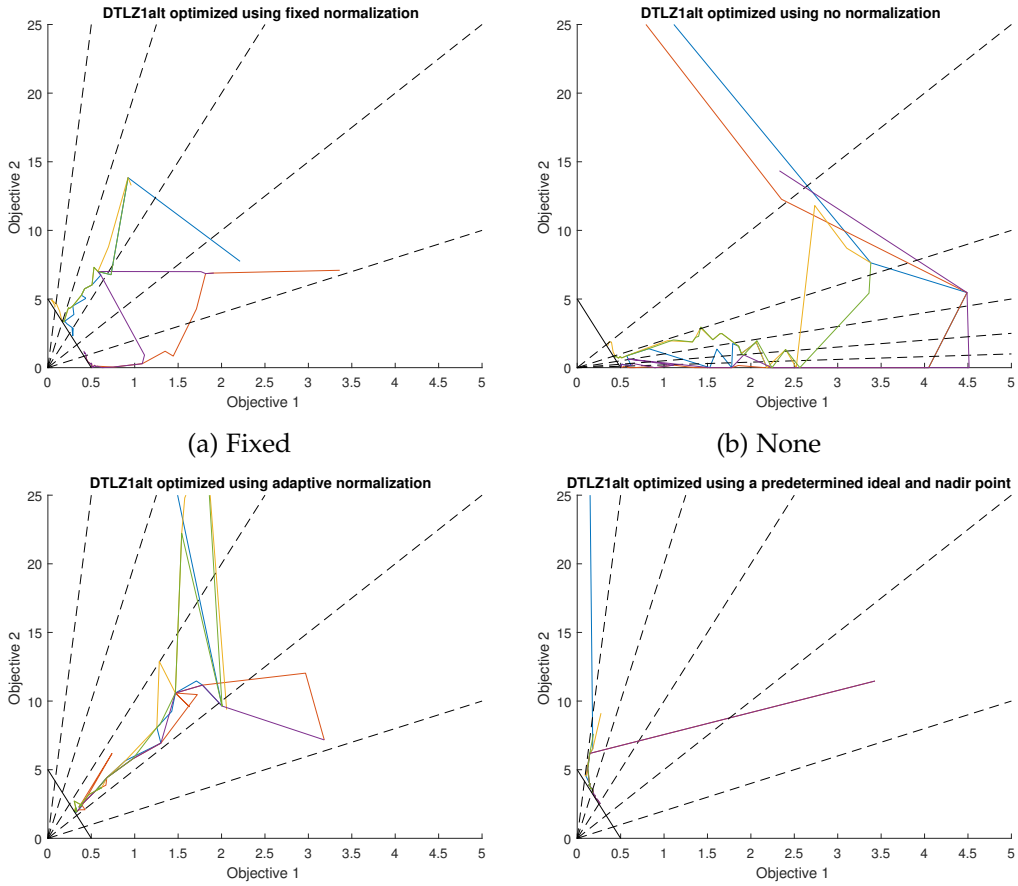


Figure 3.5: Boxplot showing how normalisation affects IGD performance. Of interest is how reserving a portion of the budget for estimated ideal and nadir points greatly improves performance.

are similar to those obtained when using no normalisation. Adaptive normalisation possesses a large variation in results, ranging from similar performance to fixed optimization to being much worse than any other method.

The last method considered was to pre-determine the ideal and nadir points by using a lexicographic optimization methodology, before commencing optimization, referred to as the pre estimate. The lexicographic optimization methodology works by imposing your preference through ordering the objective functions according to their importance or significance. In this work when finding the ideal value for an objective all importance is given to it and the other objective is ignored. When searching for the nadir the search begins for a chosen objective where the ideal search for that objective ended. It is then run such that it maintains the found ideal value of the first objective, while trying to minimise the value of the second objective. Through previous testing it was determined that 80 evaluations were needed for each objective in order to estimate an acceptable ideal point and a further 40 were needed for each to get a suitable nadir point. The median values found for the ideal and nadir after the search was run are [0, 0.0683] and [0.9134, 8.4708] respectively. During testing, it was determined that the improvement gained with additional evaluations did not balance the loss in performance



(a) Fixed (b) None

(c) Adaptive (d) Predetermined

Figure 3.6: Trajectory plots of the median runs using different normalisation methods, based on the integrated IGD. It is evident that lacking good estimates of the ideal and nadir points greatly compromises the ability of a decomposition-based optimizer to find the Pareto front.

due to fewer iterations of the optimizer. Going back to Figure 3.5 it can be seen that when the predetermined estimates for the ideal and nadir points are used, the integrated IGD is improved while the final IGD was superior to that of either adaptive normalization or when no normalization was used.

Figure 3.6 shows the trajectory plots of the median runs for the four methods. In each of the four subplots, the dashed lines indicate the reference directions. In (a), (b) and (d) the reference directions represent those that the optimizer is using. In (c) the reference directions shown are the best ones that could be found, as they change during each iteration for adaptive normalisation. Figure 3.6(a) shows the fixed normalisation, with the trajectories moving down towards the front in the same manner as seen before. In Figure 3.6(b), when no normalization is applied, all of the trajectories head down towards the first objective axis before starting to

progress towards the Pareto front. As the reference directions are evenly spaced across the objective space, they focus on the lower half of the Pareto front. In Figure 3.6(c) all of the trajectories move together before progressing down to the front, where they do not spread out. The final subplot, where the previously estimated ideal and nadir point are used, shows the points moving towards the front quickly before spreading out and managing to reach three of the reference directions (due to the points initialising close together, it is difficult to see their movement within this particular plot).

3.4.3 Interesting variants

There were a selection of interesting variants discovered during the process of testing. The first of which is when five offspring are used instead of just one. When implemented, it implies running the optimizer for only 20 iterations, so as to use the same number of evaluations in total as before. It can be seen in Figure 3.7 that, when compared to running the optimizer independently with one offspring, the trajectories obtained are much smoother. The erratic behaviour caused by the optimizer going back on itself is also almost entirely absent. This demonstrates how the addition of extra offspring allows for better directed convergence. When compared to the performance of using only a single offspring, equivalent median IGD was achieved.

The second interesting variant to be examined was discovered when running the optimizer with four neighbours and SBX reproduction with no update of neighbouring solutions. Using four neighbours meant that points from any of the subproblems had a chance of being selected during the reproduction step. The median result of running this setup can be seen in Figure 3.8. All of the different trajectories are pulled together, almost to a single point, before attempting to spread out. The rate at which they can separate out is negatively impacted by the use of SBX.

3.5 Discussion

It is evident that the sharing of information in a decomposition-based optimizer, through the use of neighbourhoods, between the different subproblems, can have a substantial impact on the optimizer's performance. The clearest example of this was observed when the component for updating the neighbourhood solutions was used. From the trajectory plot it was seen that the component enabled rapid improvement by jumping to a more advantageous point within the objective-space.

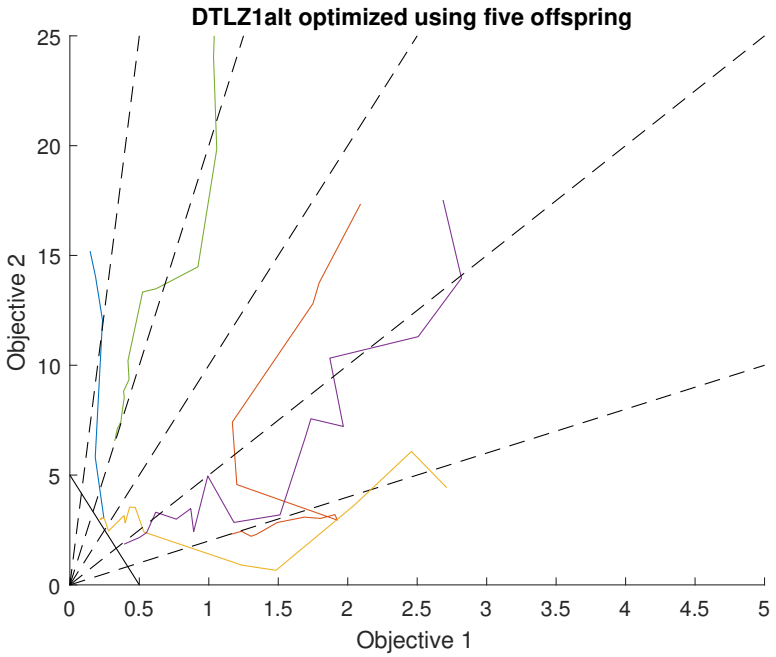


Figure 3.7: Trajectory plot when five offspring are generated, in the absence of information sharing components. Note the smoother trajectories approaching the Pareto front, when compared to the previous single offspring configurations.

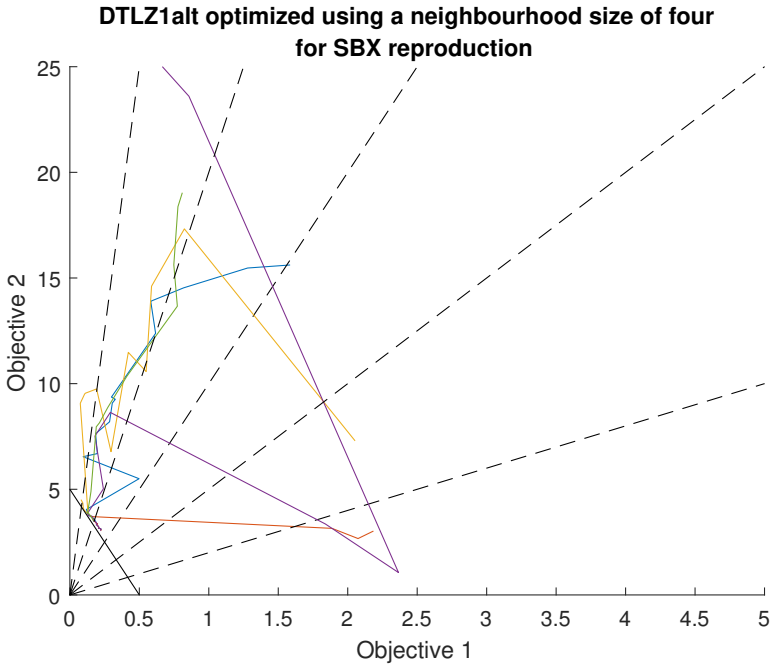


Figure 3.8: Trajectory plot when four neighbours are used for SBX and there is no update of neighbouring solutions. The points cluster together, affecting the overall ability of the optimizer to find the Pareto front.

Updating the neighbourhood solutions also allowed the solutions to spread out once they were close to the Pareto front.

While it has often been shown that the use of reproduction (or recombination) can aid optimizer performance, this appears to be invalid in the context of a small number of reference directions. The use of reproduction does initially seem to help in moving the solutions down towards the Pareto front; however once they arrive their progress comes to an almost complete stop. A likely reason for this is that while only two neighbours are being used, this constitutes a sizeable region of the Pareto front. In most problems a much larger number of reference directions are used, such as the 150 to 250 reference directions used in the original MOEA/D study (Qingfu Zhang and Hui Li, 2007). In an exploratory analysis using a 'limited' evaluation budget, the authors still used 20 reference directions and 250 iterations, which is a total evaluation budget that is an order of magnitude greater than that considered in this and other limited budget studies. Where a larger budget is available, any unhelpful effects of SBX reproduction are lessened due to the smaller distances between reference directions (see Figure 3.1). Using both SBX reproduction and the update of neighbourhood solutions together proved to be unsatisfactory, producing a worse IGD than the update of neighbouring solutions did on its own.

The results of using the different normalisation methods are also notable. Firstly, while using fixed normalisation provides the best performance, it is unrealistic for most RWAs as both the ideal and nadir points are unlikely to be known in advance. The next option of simply using no normalisation proved to be worse for the chosen disparately scaled test problem. This was due to the reference directions forcing points to one side of the front. The option of using adaptive normalisation, which is a more feasible option, proved to have a poor IGD value while also possessing a high degree of variability, which is undesirable. Using pre-estimation provided a considerable improvement over all but fixed normalisation, even though it reduced the subsequent number of evaluations available to the optimizer.

While our findings have implications for the design and configuration of decomposition-based optimizers, it is important to acknowledge that only a single problem instance was used in the analysis. Generalisation of findings to other classes of problem should be handled cautiously pending further experimental studies.

3.6 Conclusion

This chapter has demonstrated how the selection and configuration of key components of decomposition-based optimizers, represented by the seminal MOEA/D algorithm, can substantially affect the performance of the optimizer when it is being run with a limited budget of evaluations. The sharing of knowledge through the inclusion of neighbours can provide a large benefit when the update of neighbourhood solutions component is implemented. On the other hand, the inclusion of SBX reproduction when only a small number of reference directions are present, such as in the present context, is inadvisable. When considering what form of normalisation to use, it was found that using a subset of the evaluation budget to gain an estimate of the ideal and nadir points for normalisation proved more effective than the conventional adaptive scheme, and achieved close to the performance of the ideal, but infeasible, approach of knowing the correct normalisation parameters in advance. Now that an investigation into the effects of different configurations for expensive optimization has been concluded the next chapter lays out a mathematical framework for the combined problem.

Chapter 4

Framework

4.1 Introduction

In this chapter, we present a mathematical framework for laying out the combined problem of model calibration and optimization as a unified approach. This is based around the work presented in my paper *'Toward a unified framework for model calibration and optimization in virtual engineering workflows'* (Jones et al. (2019)) published in IEEE SMC. The basis for the ideas present were drawn from the papers presented within the literature review, in Sections 2.2 and 2.3.

The chapter begins with Section 4.2 by laying out the combined mathematical problem framework for model calibration and optimization. The formulation is designed to possess consistent notation through the different stages of work, which is often lacking when comparing the optimization and calibration literature. A simple demonstration of how the notation can be applied to a toy system has also been included.

The next Section 4.3 presents a selection of potential ways that such combined problems could be tackled. These concepts are expanded upon and examined in more detail within later chapters.

Section 4.4 looks at presenting a real-world virtual engineering case study within the new problem framework. The case study is taken from work done by Villarreal-Marroquín et al. (2017) on injection moulding in which both the optimization and model calibration stages were considered, although no attempt was made at laying out the problem in a unified way. This chapter finished by summarising the different conclusions in Section 4.5.

4.2 Mathematical formulation

Now that a overview of the problem has been laid out the next step it to present a mathematical formulation of the framework.

4.2.1 Problem framework and variables

We represent the physical system by

$$z_s = s(x, \psi), \tag{4.1}$$

with

- z_s: the outputs of the system;
- x: control inputs to be optimised, potentially subject to constraints;

- ψ: aleatory inputs, representing uncontrollable quantities that may differ randomly each time the physical system is run;
- $s(.,.)$: unknown function that encodes the physical laws relating inputs to outputs.

Both the inputs and outputs of the system can be multi-dimensional and are assumed to be continuous (although could be discrete). We also have a computer model of the system, represented by

$$z_m = f(x, \theta, \psi), \tag{4.2}$$

which may include additional calibration inputs θ required to 'tune' the model to the physical system. Uncertainty about θ would be epistemic.

The relationship between the model and the physical system is given by

$$s(x, \psi) = f(x, \theta, \psi) + \delta(x, \theta, \psi), \tag{4.3}$$

where $\delta(x, \theta, \psi)$ represents residual error in the model predictions, once the model has been tuned by selecting θ. The function $\delta()$ is often referred to as the model *discrepancy* (Kennedy and O'Hagan, 2001).

Toy Formula - elements of the problem

A simple toy formulation is now presented in which a ball is thrown with the aim of hitting a target. In this scenario the objective is to minimize the distance between where the ball lands and the target. It is assumed that noisy physical measurements can be obtained for the distance the ball lands from the target, and that a model of how far the ball lands from the target is also available. For illustration, we suppose that the true distance to the target (50m), the height at which the ball is thrown (2m), and the acceleration due to gravity (9.81m/s^2) are all unknown. The different components within the problem are as follows.
Model inputs:

- Control inputs: horizontal velocity (v_h) and vertical velocity (v_v)
- Calibrations inputs: distance to target (D_{ta}), starting height (H_i) and acceleration due to gravity (g)
- Aleatory input: constant horizontal acceleration exerted by wind (a)

We suppose that the physical system that described the distance from the target is

$$s(v_h, v_v, a) = |50 - D_{th}| \tag{4.4}$$

with

$$\begin{aligned} D_{th} &= (v_h t) + (0.5at^2) \\ t &= -\frac{v_v}{-9.81} + \sqrt{\left(\frac{v_v}{-9.81}\right)^2 - \left(\frac{4}{-9.81}\right)} \end{aligned} \tag{4.5}$$

The system output (to be minimised) is the distance from the target, and a constraint on the control input would be a maximum throwing speed.

We suppose that there is also an (imperfect) computer model of the system,

$$f(v_h, v_v, D_{ta}, H_i, g) = |D_{ta} - v_h t|, \tag{4.6}$$

with

$$t = -\frac{v_v}{-g} + \sqrt{\left(\frac{v_v}{-g}\right)^2 - \left(\frac{2H_i}{-g}\right)} \tag{4.7}$$

so that there will be model discrepancy resulting from the model failing to account for the wind acceleration.

Example of real world variables

Taking the more complex real world example of designing a road car, by extracting different features the variables present in the system can be organized as follows, Model inputs:

- Control Inputs - x - Driver inputs (Throttle, Break)
- Epistemic uncertainty - θ - Type of engine, vehicle body design, compressor/ torque converter spec, final drive ratio
- Aleatory uncertainty - ψ - Variation in road surface, Component temperature, NOx conversion efficiency

Model Outputs:

- Objective output - z - Speed, Fuel efficiency, Co2 emissions, Weight, Total cost

4.2.2 Model calibration

Before laying out model calibration it is important to consider what information is available for use.

Available Information

The information available for the analyst will not necessarily be the same in all problems. There are two main sources from which new data could be obtained,

either expert opinion or physical experiments. There are two suggested types of available information looked at.

- Case 1: Noisy observations. We obtain physical experimental data with observation error. For $i = 1, \ldots, n$, experiment i is carried out with control input setting x_i and measured aleatory inputs ψ_i, giving an observation

$$y_i = s(x_i, \psi_i) + \epsilon_i, \tag{4.8}$$

  where ϵ_i is an observation error.
- Case 2: Plausibility/acceptability check. An expert believes the system output would lie in some range, given the control and aleatory inputs. For $i = 1, \ldots, n$, the expert judges

$$s(x_i, \psi_i) \in S_i, \tag{4.9}$$

  for some set S_i.

In both cases it is assumed that the value of ψ_i is found. In the first case it is possible to work with an unknown ψ_i although it would increase the complexity of the problem.

Determining parameters

In model calibration, the aim is to reduce the error between the predicted and observed behaviour (Kennedy and O'Hagan, 2001). This is achieved through altering the internal parameters of the model so that it can as closely predict the 'true' outputs as possible.

Depending on what form of information is being supplied by the expert, the method needed for model calibration will vary.

- Case 1: Noisy observations
  The observations from the experiments are

$$Y = [y_1, \ldots, y_n]. \tag{4.10}$$

  If we assume a distribution for the unknown discrepancy $\delta()$ and for the errors $\epsilon_1, \ldots, \epsilon_n$, then from (4.3) we can construct a likelihood function $p(Y|\theta)$. We may then choose,

$$\underset{\theta}{\text{argmax}}\ p(Y|\theta), \tag{4.11}$$

to obtain a single best point, or, having specified a prior distribution $p(\theta)$ derive the posterior distribution,

$$p(\theta|Y) \propto p(\theta)p(Y|\theta). \tag{4.12}$$

Note that such calibration data are not just informative for θ: they would also be informative for the model discrepancy δ, so that, as in Kennedy and O'Hagan (2001), we can derive a joint posterior distribution $p(\theta, \delta|Y)$. This has important implications for optimization: via (4.3), we can attempt to optimise the physical system s, rather than the computer model approximation of it f.

- Case 2: Viability/acceptability check
  In this case there are two situations that need to be considered. The first is when it is assumed that the model matches the true system without modeling error and the second is when it does not.

  - No modeling error present
    In this case,

    $$s(x, \psi) = f(x, \psi, \theta) \tag{4.13}$$

    for some θ and all x, ψ. This gives a likelihood function,

    $$p(D|\theta) = \begin{cases} 1, & \text{if } f(x_j, \theta, \psi) \in S_i \quad i = 1, ..., N \\ 0, & \text{otherwise.} \end{cases} \tag{4.14}$$

    where D is the event of a point being accepted by an expert.
  - Modeling error present
    When using the relationship between the model and system as shown in Equation 4.3 the uncertainty due to the model error becomes an issue. One method of overcoming this is to add a tolerance to each S_i based on a judgment about $\delta(x, \psi)$. In doing this a new set S_i^* is produced which represents the original set S_i with the added tolerances. Using this the likelihood function becomes,

    $$p(D|\theta_e) = \begin{cases} 1, & \text{if } f(x_j, \theta, \psi) \in S_i^* \quad i = 1, ..., N \\ 0, & \text{otherwise.} \end{cases} \tag{4.15}$$

  In both cases it is possible to also construct a distribution for which it is necessary to find one or more values of θ that posses a $p(D|\theta) = 1$

$$p(\theta|D) \propto p(\theta)p(D|\theta). \tag{4.16}$$

4.2.3 Optimization

We suppose that the aim is to optimise the physical system, and that model discrepancy is acknowledged, so that we must consider optimising $f(x, \theta, \psi) + \delta(x, \psi)$. Were we to ignore model discrepancy, the subsequent notation could be simplified by omitting the term $\delta(x, \psi)$ throughout.

Assuming vector output quantities, we have a standard (multi-objective) optimization problem (Deb et al., 2001) written as

$$\begin{aligned} f(x, \theta, \psi) + \delta(x, \theta, \psi) =& (f_1(x, \theta, \psi) + \delta_1(x, \theta, \psi), \\ & \ldots, f_k(x, \theta, \psi) + \delta_k(x, \theta, \psi)), \end{aligned} \tag{4.17}$$

the optimization problem can be written as

$$\underset{x \in X}{\text{minimize}}\ f(x, \theta, \psi) + \delta(x, \theta, \psi), \tag{4.18}$$

subject to any constraints on x, where X is the set of possible choices of control inputs x, and minimisation is element-wise.

Pareto optimality

Working with multi-objective problems a trade-off surface between the objectives, known as the Pareto front, can be obtained by getting the set of non-dominated points from within the current population of points. A point $x \in X$ is said to be non-dominated when there does not exist a point $x' \in X$ such that

$$f_i(x', \theta, \psi) + \delta_i(x', \theta, \psi) < f_i(x, \theta, \psi) + \delta_i(x, \theta, \psi), \tag{4.19}$$

for $i = 1, \ldots, k$.

Robust optimization

Robust optimization considers optimization in the presence of uncertainty and can provide a link between the calibration and optimization stages described here. Starting with the optimization problem in (4.18), if we now consider θ, ψ and δ as uncertain, the objective function will be uncertain for any x. To recover a deterministic optimization function with a known objective function, we re-express the optimization problem as

$$\underset{x \in X}{\text{minimise}}\ I(f(x,\theta,\psi) + \delta(x,\theta,\psi)), \tag{4.20}$$

For some appropriate indicator functional I. Writing $\Omega = (\theta, \psi, \delta)$, with a corresponding sample space Ω_s, the common choices for I shown in the literature review (section 2.3.3) can be expressed as follows.

- Worst-case scenario

$$I_{wc}(x, \Omega_s) = \max_{\Omega \in \Omega_s} f(x,\theta,\psi) + \delta(x,\theta,\psi) \tag{4.21}$$

- Aggregated value

$$\begin{aligned} I_{exp}(x, \Omega_s) &= E[f(x,\theta,\psi) + \delta(x,\theta,\psi)] \\ I_{var}(x, \Omega_s) &= var[f_i(x,\theta,\psi) + \delta_i(x,\theta,\psi)] \end{aligned} \tag{4.22}$$

  where the bi-objective problem requires the selection of the minimum between the possible indicators,

$$\min_{x \in X} \left[I_{exp}(x, \Omega_s)\ ,\ I_{var}(x, \Omega_s) \right] \tag{4.23}$$

- Threshold probability

$$I_{con}(f(x,\theta,\psi) + \delta(x,\theta,\psi)), q) = p(f(x,\theta,\psi) + \delta(x,\theta,\psi) \leq q) \tag{4.24}$$

The newly determined indicator will replace the random objective function for use within an optimizer. As mentioned before there are two possible methods that can be used, either probabilistic or possibilistic. In a probabilistic approach, the calibration phase could supply the probability distribution for both θ and δ.

4.2.4 Combined workflow

Taking the components present from the calibration and optimization stages we can now summarise the combined workflow.

The physical system,

$$z_s = s(x, \psi), \tag{4.25}$$

produces expert data.

$$y = s(x, \psi) + \epsilon, \tag{4.26}$$

The model,

$$z_m = f(x, \theta, \psi), \tag{4.27}$$

relates to expert data by,

$$s(x, \psi) = f(x, \theta, \psi) + \delta(x, \theta, \psi), \tag{4.28}$$

Calibration: calculate probability of expert data given a selection of θ,

$$p(y|\theta). \tag{4.29}$$

and either select a single value, by taking the maximum likelihood,

$$\tilde{\theta} = \underset{\theta}{\operatorname{argmax}}\ p(y|\theta), \tag{4.30}$$

or produces a distribution of possible θ,

$$p(\theta|y) \propto p(\theta)p(y|\theta). \tag{4.31}$$

Then perform optimization using the discovered $\tilde{\theta}$ or by selecting $\tilde{\theta}$ from $p(\theta|y)$,

$$\underset{x \in X}{\text{minimize}}\ f(x, \tilde{\theta}, \psi) + \delta(x, \theta, \psi), \tag{4.32}$$

The robust formulation is one possible way in which the combined problem can be expressed.

4.2.5 Toy formula - The problem

Working with the same problem as before, if the ball is thrown at a particular velocity, a noisy measurement may be obtained as

$$y = s(v_h, v_v, a) + \epsilon. \tag{4.33}$$

In this case the observation error ϵ would be the difference between the measured distance the ball is from the target and the actual distance between the ball and target. The model output is expressed as,

$$z_m = f(v_h, v_v, D_{ta}, H_i, g). \tag{4.34}$$

Relating the model output to the physical measurements the expression

$$y = f(v_h, v_v, D_{ta}, H_i, g) + \delta(v_h, v_v, D_{ta}, H_i, g, a), \tag{4.35}$$

is obtained, where $\delta(v_h, v_v, D_{ta}, H_i, g, a)$ is the model discrepancy. If, for example, we wanted a point estimate of the calibration inputs $\theta = [D_{ta}, H_i, g]$, we would

need to obtain

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\ p(y|\theta), \tag{4.36}$$

which could then be used to calculate the model discrepancy.

The newly acquired set of parameters, $\hat{\theta} = (\hat{D}_{ta}, \hat{H}_i, \hat{g})$, can now be used in optimization,

$$\underset{x \in X}{\text{minimize}}\ f(x, \hat{\theta}) + \delta(x, \hat{\theta}, a), \tag{4.37}$$

for a given a, with $x = (v_h, v_v)$. $\delta()$ would initially take the form of an assumed distribution which could be improved through the addition of knowledge gained from experimentation.

4.3 Possible solutions

There are three logical routes that solutions to this problem could take. The first would be to use a robust optimization approach, without performing any calibration. This could save budget in terms of physical experiments, and potentially computer model runs, but may result in an overly-conservative result as a consequence of greater parameter uncertainty. A second option is to perform calibration before beginning the optimization process. This may reduce the risk of a conservative solution, but at a greater cost in terms of the computation budget. A third option is to alternate between performing optimization and model calibration based upon a chosen criteria. This may be beneficial if model discrepancy is a concern: as the non-dominated set of control inputs is reduced, further calibration experiments can be performed, which may improve estimates of model discrepancy precisely in the regions of interest within the control input space.

Figure 4.1 provides a sketch of how such a third option could be realised. The processes of calibration and optimization are alternated, with the transition between the two determined by switching conditions. These conditions could be based on number of evaluations (or proportion of the budget), or use convergence criteria that are relevant to either calibration or optimization (e.g. respectively, robustness of optimal output to parameter uncertainty or negligible improvement in a hypervolume indicator (Zitzler et al., 2007)).

A further area to consider is the use of low-complexity *surrogate models* (also known as *emulators* or *meta-models*) to replace expensive evaluations within either the calibration process, optimization process, or both processes (Chugh et al., 2019, Kennedy and O'Hagan, 2001). Any of the above approaches can incorporate surrogates. Surrogates have the further benefit that they can use information from both the calibration and optimization runs, allowing for greater information sharing

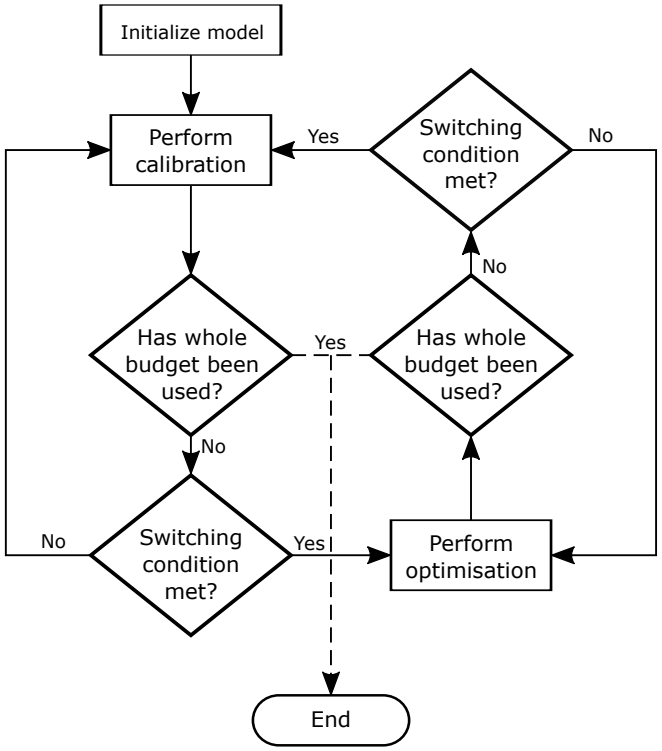


Figure 4.1: Flow diagram for the proposed joint problem of model calibration and optimization.

between the two activities.

4.4 Real-world examples set within the combined problem framework

This section sets out an example problems within the combined problem framework for optimization and model calibration. The problem presented is from a paper on injection molding (Villarreal-Marroquín et al., 2017).

4.4.1 Injection moulding

The methodology used within the injection moulding paper is linear with a fixed number of steps which are each only undertaken once. The basic workflow as presented within the paper is as follows,

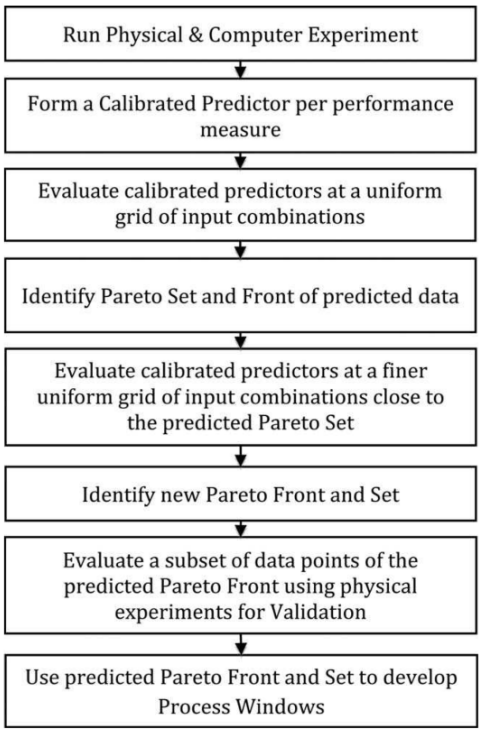


Figure 4.2: Injection molding process optimization and calibration framework taken from the original paper (Villarreal-Marroquín et al., 2017).

Within the paper the method was broken down into eight stages which mostly corresponds to the workflow above. There are slight deviations from Figure 4.2 however as process windows are not explicitly developed. Returning to the problem it is possible to extract the components present within the system and obtain the,
Model inputs:

- Control Inputs - Melting temperature (°C), Packing time (s), Packing pressure (MPa), Cooling time (s),
- Epistemic uncertainty - HTC flow ($W/m^2\ k$), HTC pack ($W/m^2\ k$), HTC open ($W/m^2\ k$),

Model Outputs:

- Objective output - Relative shrinkage of the length (L3), relative shrinkage of the thickness (T3) and relative shrinkage of the width (W3).

The objective is to choose control inputs that will minimise all three outputs. There are no aleatory inputs in this example.

Their methodology proceeds in a series of stages. We give a condensed list here, skipping those stages that are not directly relevant to our framework.

Stage 1 - Design of experiment

The methodology starts by acquiring two sets of data. The first set consisting of physical observed values at 19 different settings of control inputs, each replicated four times. These initial control inputs ($\check{\mathbf{x}}$) are determined using a full factorial with n=2 (i.e taking the high and low values of all combinations). The second set of data comes from running a computer simulation of the process. The initial control inputs and parameters for the simulation are determined through the use of the Maximin Latin Hypercube design (obtained from URL{https://spacefillingdesigns.nl}) and are represented by $\mathbf{x}_{hyp}$ and $\boldsymbol{\theta}_{hyp}$ respectively. It is not mentioned which method was used for the creation of the maximin design they chose to use from the database.

The physical data comes from,

$$\hat{y} = \frac{1}{N}\sum_{i=1}^{N}(y_s(\check{\mathbf{x}}_i)), \tag{4.38}$$

where the average of the four repetitions is being taken for the final value. This design is known as a trend free split-plot design. The simulated data is obtained using the model function,

$$z_m = f(\mathbf{x}_{hyp}, \boldsymbol{\theta}_{hyp}). \tag{4.39}$$

Stage 2 - Fit calibrated predictor

The computer model is computationally expensive, and so the authors do not use it 'directly' for either calibration or optimization. They also recognise the presence of model discrepancy, which they account for in their optimization. This is done through the construction of a 'calibrated predictor'. Using the two datasets (physical experiments and computer model runs), they

- construct a meta-model of the computer model, based on Gaussian process regression;
- construct an estimate $\hat{\delta}(x, \theta)$ of the model discrepancy between reality and the simulation;
- derive a posterior distribution $p(\theta|\hat{\mathbf{y}})$ of the calibration inputs θ given the two datasets.

The calibrated predictor takes the form

$$\hat{s}(x) = \mathbb{E}_\theta[\hat{f}(x, \theta)] + \hat{\delta}(x, \theta), \tag{4.40}$$

where $\hat{f}$ is an estimate of the computer model obtained from the meta-model, and the expectation is taken with respect to the posterior distribution of the calibration inputs θ.

Once this has been completed the calibrated Gaussian predictors are then evaluated at the same control input settings as the initial physical experiments to allow for a comparison of results. The root mean square prediction error (RMSPE) is used to assess the accuracy of the calibrated predictor.

$$RMSPE = \sqrt{\frac{\sum_{i=1}^{j}(\check{f}(\check{\mathbf{x}}_i) - y_s(\check{\mathbf{x}}_i))^2}{j}}, \tag{4.41}$$

where j is the number of physical experiments that have been carried out.

Stage 3 - First attempt to optimise the physical system

The authors perform optimization by constructing a grid of equally spaced points, $\mathbf{x}_g$ spanning each of the control inputs. In their case study, the authors used a grid of 360 points. Once the grid is constructed, the next step is to evaluate (4.42) at each grid point (producing three outputs per grid point),

$$\hat{s}_g(\mathbf{x}_{gi}) = \mathbb{E}_\theta[\hat{f}(\mathbf{x}_{gi}, \theta)] + \hat{\delta}(\mathbf{x}_{gi}, \theta), \tag{4.42}$$

where i goes from one to the number of grid points. It should be noted that they do not state what summary statistic they were using on z.

Stage 4 - Obtain Pareto front

A Pareto front is obtained by inspection of the grid. The condition for selecting the non-dominated points that comprise the Pareto front is provided in equation 4.19. Given the averaging with respect to θ, this corresponds to the robust optimization procedure with the functional I chosen to be the expectation operator.

Stage 5 - Refine Pareto front

Having identified (approximately) the Pareto front, a second grid of control inputs is chosen in the location of the front, and (4.42) is again evaluated at each grid point. A refined Pareto front is then identified by inspection. The authors used 560 grid points in their case study.

Stage 8 - Validate final front

The estimated Pareto front has been obtained from a 'calibrated predictor', evaluated at 920 settings of the control inputs. The calibrated predictor is only an estimate of the physical system, based on 19 (replicated) physical experiments, and 35 computer model runs. Consequently, there is a need to validate the estimated Pareto front. This is done by selecting five control input settings from the front, and then performing physical experiments at these settings. The system outputs are then compared with those from the initial 19 physical experiments. No replications of the validation run physical values are carried out.

Extension

Although the methodology that is described only performs two loops for optimization it would be relatively easy to extend this work to expand it for more iterations. To actually implement this all that would be needed is to repeat step 5 for as many times as necessary. An alternative to increasing the number of iterations would be to use a finer grid of points. Both of these options should aid in improving the results obtained, however, incorporating either would increase the number of evaluations of the computer simulation that would be have to be run.

Critique

After going through the paper there are some areas that seem to be lacking or unclear. The main points identified were,

- Some stages are not clear on what is being done and some of the information being used is not provided to the reader.
- There is no consideration of structural uncertainty.
- There is no comparison to more advanced optimization strategies.
- Limitations of this method are not discussed in detail.
- The predictors are not correlated.

These make it harder to determine whether this setup would be viable for a larger array of problems and are aspects that will need to be taken into consideration during future work.

4.5 Conclusion

This chapter has set out a mathematical notation that is consistent for problems within which both calibration and optimization parameters are present. Examples

of how problem components can be manipulated to fit within such a framework have been demonstrated. Possible methods that could be used for approaching such a problem have been presented and their viability commented upon.

The proposed formulation has then been used as a basis to lay out a real world problem possessing both control inputs as well as model parameters. The example was drawn from a practical study examining the calibration and optimization of the injection moulding process. From this example it can be seen that this formulation is capable of adequately including the features present within such problems.

The fact that the problem framework is capable of being used for defining a diverse set of different problems poses many benefits. One such benefit is that provided a method is developed to work with this framework, any problem capable of being defined using this framework should be compatible with the new method. Another benefit of having such a combined problem framework is that it should be easier to extract and categorise information.

Now that a viable mathematical formulation combining the two areas of interest form the virtual engineering workflow (VEW) has been produced. The next step is to lay out a methodology for benchmarking problems of this new setup. The following chapter looks at the performance metrics which could be used with the new problems before going on to lay out possible benchmarking problems and finally constructing a new component to work with the pre-existing WFG benchmarking framework.

Chapter 5

Benchmarking

5.1 Introduction

This chapter looks at benchmarking, which is the process by which the performance of a methodology can be assessed. In order for benchmarking to be carried out it is necessary to set it up as a systematic structured process. Within this process there are several components that need to be selected. The two examined here are the performance metric and the test problem. The first of these to be presented is the possible options that could be utilised as the performance metric. Those looked at include the hypervolume indicator, inverted generational distance and the epsilon family.

The chapter then moves on to examine possible test problems. In order to do this, it is first necessary to come up with a method for developing test problems that possess both model parameters and control inputs. The proposed approach involves altering pre-existing test problems from within the optimization literature and was carried out to obtain a selection of single and multi-objective problems.

The third area the chapter details is the creation of a new component for the Walking Fish Group (WFG) framework. This was found to be necessary as a problem which could not be solved, by a simple surrogate model, was desired. The WFG framework was selected due to its modularity, this feature means that the newly developed component can fit within a wide range of setups and achieve a variety of properties.

The chapter follows the areas laid out above with the performance indicators being shown in Section 5.2, the adapted optimization test problems presented in Section 5.4 and finally the development of the new component being described in Section 5.5. In addition to these a brief overview of COmparing Continuous Optimisers (COCO) and Black Box Optimization Benchmarking (B-BOB) is presented in Section 5.3.

5.2 Indicators

There are many different indicators, also known as performance metrics, present within the literature as shown by Riquelme et al. (2015) in their paper on performance metrics for multi-objective optimization. Their assessment of performance metrics focused around those used for Evolutionary multi-criterion optimization (EMO). They present that between 2005 and 2013 within the literature on EMO approximately 53% of articles used performance indices and of these about 43% chose to use only a single metrics. This means that during this period only just over 30% of article used more than one performance metric. Out of the perfor-

mance metrics used Hypervolume was by far the preferred choice, followed up by Generational distance and the Epsilon family. Each of these methods shall now be discussed in addition to that of the inverted generational distance.

5.2.1 Hypervolume

The hypervolume indicator works by calculating the area between the current undominated points and an anti-ideal point. A comprehensive overview of the hypervolume indicator is given by (Bader, 2010). It possesses the advantage of giving a representation of both the accuracy and the diversity on the discovered points as well as being capable of working with any number of objectives. One issue with using the hypervolume is the complexity of calculating it, while it is possible to use for high dimensions, as the number of objectives increases so does the computational difficulty and the time required. An example of the hypervolume can be seen if Figure 5.1 with the hypervolume area being marked by a dashed line. It can be clearly seen from the graph that if either the known undominted points moved further from the anti-ideal or new undominated points were to be found then the size of the hypervolume would increase.

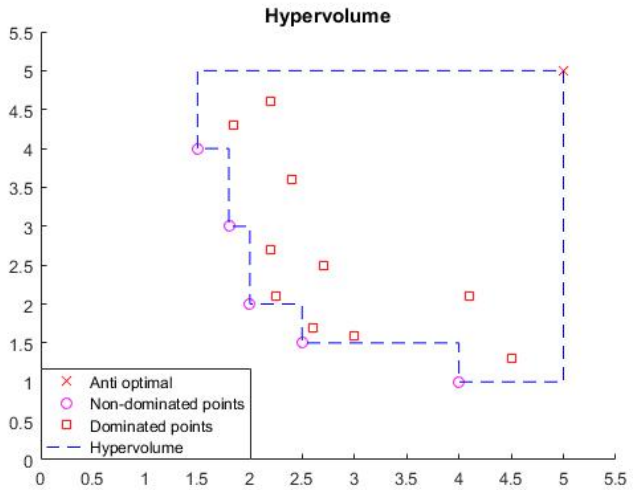


Figure 5.1: Example of hypervolume, with non dominated points shown as circles, the selected anti ideal shown as a cross and the Hypervolume area encircled by a dashed line

The hypervolume does not require knowledge of the Pareto front in order to be useful as it is possible to take arbitrary points for the ideal and anti-ideal. If this is done, then it is possible to tell whether the performance of the optimizer is improving, although it is not however possible to determine how close the results

of the optimizer are to the optimal outputs. When testing an optimizer on a pre-known model the hypervolume has the added benefit of informing the decision maker how close to the optimizer is to the best achievable performance. In order to calculate this the hypervolume it calculated using the Pareto front and anti-optimal point. The optimizer can then be run while trying to minimize the difference between this ideal hypervolume and that obtained from the non-dominated point set. The hypervolume does suffer from bias being present when a reference point is badly chosen. Zitzler et al. (2007) look at modifying it in order to obtain other preferences. They determined that it is possible to incorporate other preferences while managing to maintain the hypervolume indicators sensitivity to the domain.

5.2.2 Generational Distance

Generational distance looks at the average of the Euclidean distances between the points in the current best Pareto front and the nearest point from the true Pareto front to each of them (Van Veldhuizen and Lamont, 1998). While generational distance does inform the decision maker of how close to the Pareto front the non-dominated set is it requires the user to have knowledge of the front's location and so does not work with arbitrary test problems. From this, it can be taken that obtaining a Generational distance of zero would mean that all points in the calculated Pareto front lie on the true front. The Large disadvantage of this method is that it gives no information about the spread of points over the true Pareto front.

5.2.3 Inverted Generational Distance

Inverted generational distance (Ishibuchi et al., 2015) works similarly to that of Generational distance. Once again, it is necessary to have a set of reference points representing the true Pareto front as well as the set of points representing the current best set of points. The difference in methods is that for the inverted generational distance the Euclidean distance from each point in the true Pareto front set and the closest point from the set of current best points is averaged. The formula for calculating the inverted generational distance with Pareto front reference point set $Z = (z_1, z_2, ...z_{|Z|})$ and 'best' solution set $Y = (y_1, y_2, ..., y_{|Y|})$ is:-

$$IGD = \frac{1}{|Z|} \sum_{j=1}^{|Z|} \min_{y_i \in Y} d(y_i, z_j) \tag{5.1}$$

Through looking at the distances from each point in the true Pareto front, it becomes possible to assess the spread of points over the true front. The require-

ment of points needing to be spread over the Pareto front makes it a harder metric to meet.

5.2.4 Epsilon family

There are two main Epsilon methods, the epsilon indicator and additive epsilon indicator (Zitzler et al., 2002). Both methods function in the same manner with the only difference being how ϵ is applied. In the epsilon indicator a value of ϵ is determined to represent an approximate value with respect to all directions, where one set of objectives is worse than another. In the additive epsilon indicator, ϵ represents an amount that needs to be added instead of a factor. Assuming there are two objective vectors $Y^1 = (y_1^1, y_2^1, ..., y_{|Y^1|}^1)$ and $Y^2 = (y_1^2, y_2^2, ..., y_{|Y^2|}^2)$ the binary ϵ -indicator is determined by:-

$$I_e(A, B) = \max_{y^2 \in B} \min_{y^1 \in A} \max_{1 \leq i \leq n} \frac{y_i^1}{y_i^2} \tag{5.2}$$

for any two approximation sets A,B (Zitzler et al., 2002). The additive binary ϵ -indicator is determined by(Bringmann et al. (2014)):-

$$I_e(A, B) = \max_{y^2 \in B} \min_{y^1 \in A} \max_{1 \leq i \leq n} y_i^2 - y_i^1 \tag{5.3}$$

5.3 COCO / B-BOB framework

COmparing Continuous Optimisers (COCO) is a process that has been developed in order to give users both a experimental framework as well as a method for processing the results of benchmarking for clear presentation (Hansen et al., 2011). The main team behind the development of the software that is used in COCO is TAO. COCO is a popular method for performing Black Box Optimization Benchmarking (B-BOB) having been used for around 100 articles and algorithms. COCO and the B-BOB methodology gives a standardised method for analysing the performance of optimizers. This allows for them to be more easily compared. The methodology for performing B-BOB is available for everyone to see and has clear reasoning behind its implementation choices

5.4 Adapting test problems from the literature

5.4.1 Examples of possible test functions

Initially a large selection of test problems were considered from the optimization literature, see 5.1, from which a small selection were chosen. Within these problems taken from the optimization literature, the main structure of the functions have been kept the same while constants have been changed into model parameters that are now required to be found. Each problem requires two setups, one without model error and one where the structure of the function has been altered so that an error exists between the two model setups. These are to simulate the real world physical system and an imperfectly model of that system.

5.4.2 Proposed method for adapting test problems

Initially the prospect of directly adapting test problems from within the literature was examined.

When creating the test problems like this a pre existing problem from the literature was selected as a starting point. This was done as they are what other within the field of optimization are currently familiar with. Each of them have known characteristics and shapes which have been set as the ideal setup of the function. The problems where first taken and possible parameters selected. These were broken down to two groups, those which change the modelled front and those which change the inputs required to gain that front. Those belonging to the first category cause a discrepancy between what is observed from the model and the system while those of the second prevent the optimal inputs being obtained while their true values have not been found.

In most cases these parameters are simply constants that were already present within the function which have been changed to variables. A range of possible values surrounding the initial constant value were selected and used as the parameter range. If possible the range has been chosen so that any repeating patterns are avoided. Once this is done the, 'modelled' version of the function is produced through altering it in some way. Currently the functions are changed so that the fronts have an altered shape. In the case of the WFG functions this is very simple to achieve due to their modular framework.

5.4.3 Single objective

The Rastrigin and Schwefel function have been chosen as examples of possible single objective problems.

Name	No Objectives	No Inputs	Separability	Unimodal	Geometry
ZDT1	2	2+	S	Y	convex
ZDT2	2	2+	S	Y	concave
ZDT3	2	2+	S	[Y N]	disconnected
ZDT4	2	2+	S	[Y N]	convex
DTLZ1	2+	2+	S	N	linear
DTLZ2	2+	2+	S	Y	concave
DTLZ3	2+	2+	S	N	concave
DTLZ4	2+	2+	S	Y	concave
DTLZ5	2+	2+		Y	
DTLZ6	2+	2+		Y	
DTLZ7	2+	2+	[NS NS ... S]	[Y Y ... N]	disconnected
MOP1	2	1	S	Y	convex
MOP2	2	3+	S	Y	concave
MOP3	2	2	[NS S]	[N Y]	disconnected
MOP4	2	3+	S	[Y N]	disconnected
MOP5	3	2	NS	[N Y N]	
MOP6	2	2	S	[Y N]	disconnected
MOP7	3	2	[S NS NS]	Y	disconnected
WFG1	2+	24	S	Y	convex, mixed
WFG2	2+	24	NS	[Y N]	convex, disconnected
WFG3	2+	24	NS	Y	Linear, degenerate
WFG4	2+	24	S	N	concave
WFG5	2+	24	S		concave
WFG6	2+	24	NS	Y	concave
WFG7	2+	24	S	Y	concave
WFG8	2+	24	NS	Y	concave
WFG9	2+	24	NS	N	concave
BK1	2	2	S	Y	convex
DGO1	2	1	NA	N	convex
DGO2	2	1	NA	Y	point
FF1	2	2	S	Y	concave
JOS1	2	2+	S	Y	convex
JOS2	2	2+	[NA S]	Y	mixed
LRS1	2	2	S	Y	convex
LE1	2	2	S	Y	concave
MLF1	2	1	NA	N	convex
QV1	2	2+	S	N	concave
SP1	2	2	NS	Y	convex
VU1	2	2	S	Y	convex
VU2	2	2	S	Y	convex

Table 5.1: A list of some of the available test problems from within the literature indicating their, number of objectives and inputs, if they are separable or unimodal and what shape geometry they posses.

Rastrigin function

$$\begin{aligned} &\text{Minimize } f(x) = \theta_1 n + \sum_{i=1}^{n} [x_i^2 - \theta_1 cos(\theta_2 x_i)] \\ &\quad \text{where } x_i \in [-5.12, 5.12]. \end{aligned} \tag{5.4}$$

where the model inputs are,

$$Inputs = [x_1 \ x_2 \ ... \ x_n] \tag{5.5}$$

and the model parameters are,

$$Parameters = [\theta_1 \ \theta_2] \tag{5.6}$$

The values for the inputs at the optimal solution are $x = (0, 0, ..., 0)$ when 'true' model parameters $\theta_1 = 10$ and $\theta_2 = 2\pi$ are used.

Schwefel Function

$$\begin{aligned} &\text{Minimize } f(\mathbf{x}) = f(x_1, x_2, ..., x_n) = \theta_1 n - \sum_{i=1}^{n} \theta_2 x_i sin\left(\sqrt{|x_i| + \theta_3}\right) \\ &\quad \text{where } x_i \in [-500, 500]. \end{aligned} \tag{5.7}$$

where the model inputs are,

$$Inputs = [x_1 \ x_2 \ ... \ x_n] \tag{5.8}$$

where the model parameters are,

$$Parameters = [\theta_1 \ \theta_2 \ \theta_3] \tag{5.9}$$

The values for the inputs should be $x = (420.9687, ..., 420.9687)$ when $\theta = (418.9829, 1, 0)$

5.4.4 Multi objective

DTLZ1$_\theta$

A less rugged version of the DTLZ1 (Deb et al., 2005) function obtained from the Parego paper (Knowles, 2006). It originally possessed a straight Pareto front

that went between [0.5 0] and [0 0.5].

$$\begin{aligned}
&\text{Minimize } f_1 = \theta_1 x_1(1+g)\\
&\text{Minimize } f_2 = \theta_2(1-x_1)(1+g)\\
&\qquad g = 10\Big[\theta_3 + \sum_{i\in\{2,\dots,6\}} (x_i+\theta_4)^2 - cos(2\pi(x_i+\theta_5))\Big]\\
&\quad \text{where } x_i \in [0,1], i \in \{1,\dots,n\}, n = 6.
\end{aligned} \tag{5.10}$$

When model error is included the structure of the function changes to become,

$$\begin{aligned}
&\text{Minimize } f_1 = \theta_1 x_1(1+g)\\
&\text{Minimize } f_2 = \theta_2(1-x_1^2)(1+g)\\
&\qquad g = 10\Big[\theta_3 + \sum_{i\in\{2,\dots,6\}} (x_i+\theta_4)^2 - cos(2\pi(x_i+\theta_5))\Big]\\
&\quad \text{where } x_i \in [0,1], i \in \{1,\dots,n\}, n = 6.
\end{aligned} \tag{5.11}$$

where the model inputs are,

$$Inputs = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6], \tag{5.12}$$

and the model parameters are,

$$Parameters = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5]. \tag{5.13}$$

To obtain the Pareto front the inputs should be set to $x_m = 0.5$ for all but x_1 which should be in the range of $[0, 1]$. The parameters that correspond to the 'true' model are $\theta = (0.5, 0.5, 5, -0.5, -0.5)$. The Model error that has been incorporated causes the shape of the front to become concave. Each of the parameters impact the model in different ways, θ_1 and θ_2 work to scale the objectives, θ_3 represents an amount that is required to cancel out the effects of inputs $x_2 : x_n$. These first three parameters will not have any effect on the inputs required to achieve the 'true' Pareto front, they simply alter the output of the model. θ_4 has a varying impact which depends on the difference between the current input and the value of θ_4 when this difference is small it has a negligible impact on what constitutes a good input. θ_5 has the largest impact on which inputs will produce the 'true' Pareto front, when θ_4 is a reasonable approximation of its true value the inputs $(x_2 : x_n)$ that will make up the Pareto front are $x_i = -\theta_5$.

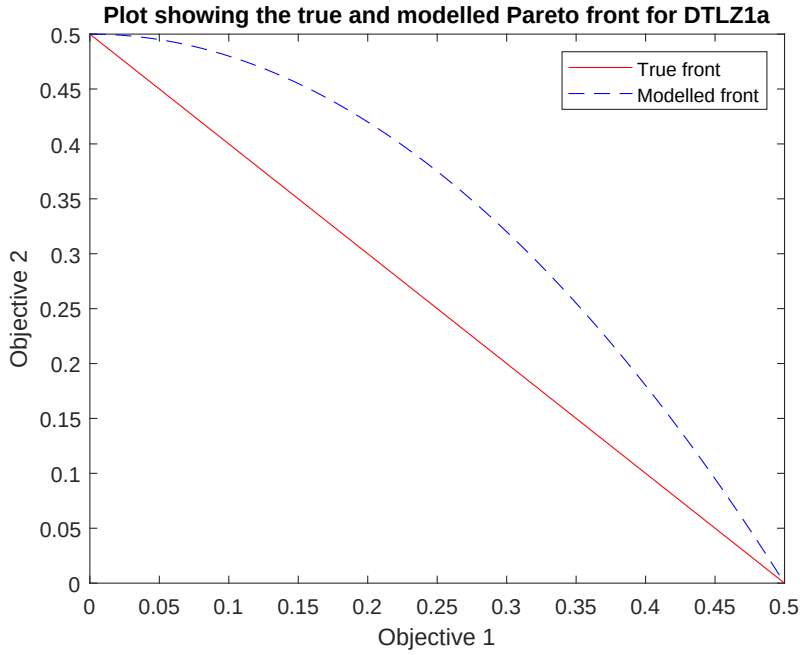


Figure 5.2: A plot showing both the true Pareto front as well as that produced by the model containing a structural error for the DTLZ1a function.

ZDT1$_\theta$

ZDT1 (Zitzler et al., 2000) originally possessed a convex Pareto front that went between [1, 0] and [0 1].

$$\begin{aligned}
&\text{Minimize } f_1(x) = x_1 \\
&\text{Minimize } f_2(x) = g(x)h(f_1(x), g(x)) \\
&\qquad g(x_2, ..., x_m) = \theta_1 + \theta_2 \Big[\sum_{i=2}^{m} (x_i) \Big] \\
&\qquad h(f_1, g) = 1 - \sqrt{f_1/g} \\
&\text{where } x_i \in [0, 1], m = 30.
\end{aligned} \tag{5.14}$$

When model error is included the structure of the function changes to become,

$$\begin{aligned}
&\text{Minimize } f_1(x) = x_1 \\
&\text{Minimize } f_2(x) = g(x)h(f_1(x)) \\
&\qquad g(x_2, ..., x_m) = \theta_1 + \theta_2\Big[\sum_{i=2}^{m}(x_i)\Big] \\
&\qquad h(f_1) = 1 - f_1 \\
&\text{where } x_i \in [0,1], m = 30.
\end{aligned} \tag{5.15}$$

where the Model inputs are

$$Inputs = [x_1\ x_2\ ...\ x_{30}] \tag{5.16}$$

and the Model parameters are

$$Parameters = [\theta_1\ \theta_2] \tag{5.17}$$

Similarly to the DTLZ1 function to obtain the true Pareto front all but the first input need to have the same value, in this case x=0. Once again the first input, x_1 , indicates direction and can takes a value between [0, 1]. The 'true' parameters that match the original problem are $\theta = (1, 9/29)$. Both θ_1 and θ_2 affect the distance between a point and the origin while also having some impact on the direction

WFG2$_\theta$

The WFG2 test problem comes from the walking fish group test suite presented by Huband et al. (2005) and originally possessed a convex disconnected Pareto front. Its main components consist of,

Shape	$h_{m=1:M-1}$=$convex_m$
	h_m=$disc_m$ (with $\alpha = \beta = 1$ and $A = \theta_1$)
t1	$t^1_{i=1:k}$=y_i
	$t^1_{i=k+1:n}$=$s\_linear(y_i, \theta_2)$
t2	$t^2_{i=1:k}$=y_i
	$t^2_{i=k+1:k+l/2}$=$r\_nonsep(\{y_{k+2(i-k)-1}, y_{k+2(i-k)}\}, 2)$
t3	$t^3_{i=1:M-1}$=$r\_sum(\{y_{(i-1)k/(m-1)+1}, ..., y_{ik/M-1)}\}, \{1, ..., 1\})$
	t^3_M=$r\_sum(\{y_{k+1}, ..., y_{k+l/2}\}, \{1, ..., 1\})$

When model error is included the structure of the function changes to become,

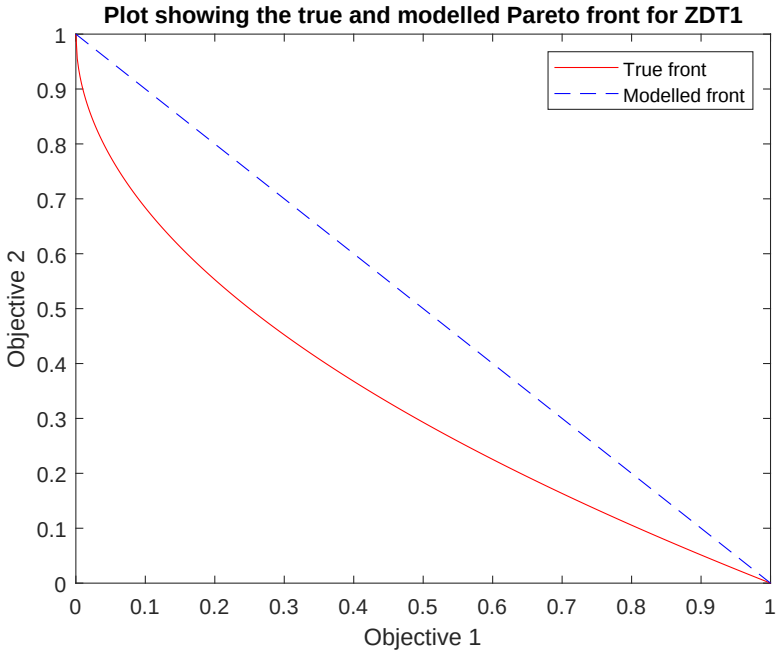


Figure 5.3: A plot showing both the true Pareto front as well as that produced by the model due to model error for the ZDT1$_\theta$ function.

$$
\begin{array}{ll}
\text{Shape} & h_{m=1:M-1}=concave_m \\
 & h_m=disc_m \text{ (with } \alpha=\beta=1 \text{ and } A=\theta_1) \\
\text{t1} & t^1_{i=1:k}=y_i \\
 & t^1_{i=k+1:n}=s\_linear(y_i,\theta_2) \\
\text{t2} & t^2_{i=1:k}=y_i \\
 & t^2_{i=k+1:k+l/2}=r\_nonsep(\{y_{k+2(i-k)-1},y_{k+2(i-k)}\},2) \\
\text{t3} & t^3_{i=1:M-1}=r\_sum(\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/M-1)}\},\{1,\ldots,1\}) \\
 & t^3_M=r\_sum(\{y_{k+1},\ldots,y_{k+l/2}\},\{1,\ldots,1\})
\end{array}
$$

where the Model inputs are,

$$Inputs = [x_1\ x_2\ \ldots\ x_{24}] \tag{5.18}$$

and the Model parameters are,

$$Parameters = [\theta_1\ \theta_2] \tag{5.19}$$

For the WFG2$_\theta$ problem the first four inputs control direction while the last 20 alter the distance. In this case the Pareto front is found when x=0.35i, where i is the current input being looked at, for all distance inputs. The possible range of inputs are between zero and 2i (this can also be shown as $2:2:2n$) and the

'true' parameters in this case are $\theta = (5, 0.35)$. The effect of θ_1 is to change the number of disconnected regions making up the pareto optimal front. The second parameter, θ_2, is highly important is it determines the value for which the control inputs achieve an optimal performance.

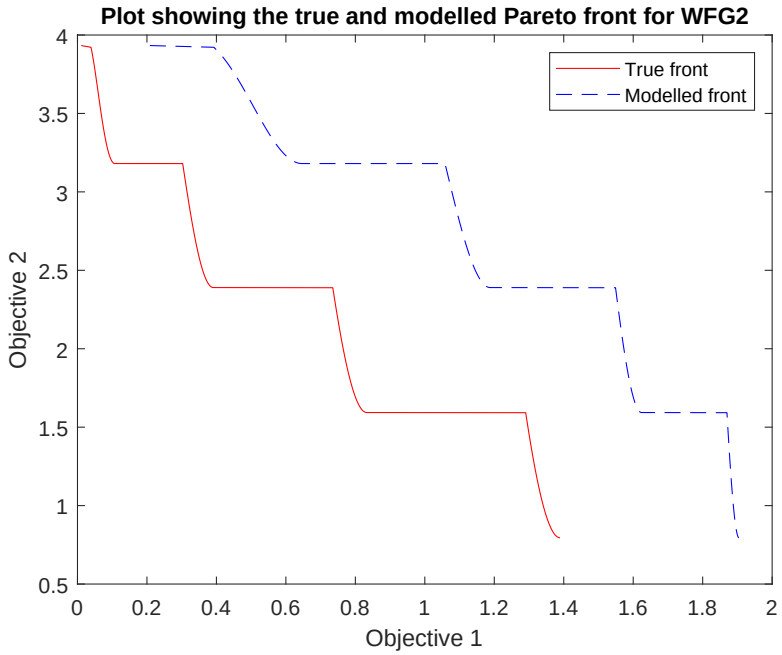


Figure 5.4: A plot showing both the true Pareto front as well as that produced by the model due to model error for the WFG2 function.

WFG4$_\theta$

The WFG4 test problem also comes from the comes from walking fish group test suite and originally possessed a convex disconnected Pareto front. Its main components consists of,

$$
\begin{aligned}
&\text{Shape} && h_{m=1:M}=concave_m \\
&\text{t1} && t^1_{i=1:n}=s\_multi(y_i, 30, \theta_1, \theta_2) \\
&\text{t2} && t^2_{i=1:M-1}=r\_sum(\{y_{(i-1)k/(m-1)+1}, ..., y_{ik/M-1)}\}, \{1, ..., 1\}) \\
& && t^2_M=r\_sum(\{y_{k+1}, ..., y_n\}, \{1, ..., 1\})
\end{aligned}
$$

When model error is included the structure of the function changes to become,

$$
\begin{aligned}
&\text{Shape} && h_{m=1:M}=linear_m \\
&\text{t1} && t^1_{i=1:n}=s\_multi(y_i, 30, \theta_1, \theta_2) \\
&\text{t2} && t^2_{i=1:M-1}=r\_sum(\{y_{(i-1)k/(m-1)+1}, ..., y_{ik/M-1)}\}, \{1, ..., 1\}) \\
& && t^2_M=r\_sum(\{y_{k+1}, ..., y_n\}, \{1, ..., 1\})
\end{aligned}
$$

where the Model inputs are,

$$Inputs = [x_1\ x_2\ ...\ x_{24}] \tag{5.20}$$

and the Model parameters are,

$$Parameters = [\theta_1\ \theta_2] \tag{5.21}$$

For the WFG4$_\theta$ problem the first four inputs control direction while the last 20 alter the distance. In this case the Pareto front is found when x=0.35i, where i is the current input being looked at, for all distance inputs. The possible range of inputs are between zero and 2i (this can also be shown as $2:2:2n$). The 'true' parameters in this case are $\theta = (10, 0.35)$. The first model parameter, θ_1, controls the magnitude of the multi-modality present within the problem. Similarly, to the WFG2$_\theta$ problem the second parameter, θ_2, determines the value for which the control inputs achieve an optimal performance.

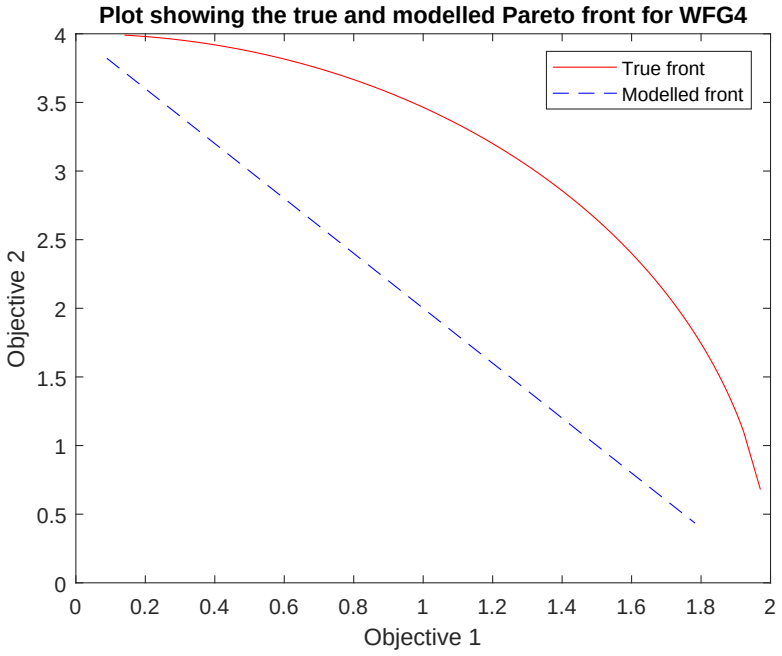


Figure 5.5: A plot showing both the true Pareto front as well as that produced by the model due to model error for the WFG4 function.

5.5 Creation of a new test problem

5.5.1 Requirements

There are certain characteristics that are desired from the test problem, these include,

- Non-separable
- Scalable
- Adjustable complexity
- Can not be easily approximated via a simple surrogate model

The main points that are not currently addressed satisfactorily within the literature are the ability to adjust the complexity of a problem and the lack of problems that are tricky to approximate through the use of a simple function. In regards to the second issue, in a lot of cases problems have been produced via taking an initial shape such as x^2 and applying complexity to it with a function such as $sin(x)$. Such a problem can be seen within Figure 5.6.

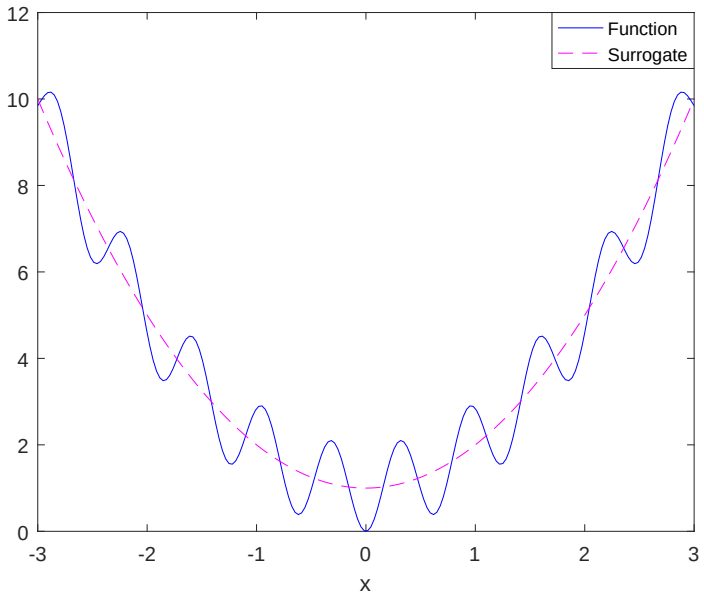


Figure 5.6: Example of how a simple surrogate can fit to a more complex function

While such problems may be difficult to solve normally when applying a surrogate the solution can often be easily found. In this case even though the surrogate does not match up to the true function it was still possible to determine

the optimal value. In regards to adjusting the complexity of a problem the WFG toolkit already allows for this to some extent in addition to allowing problems to be built up from components. Due to these aspects it seems sensible to make a new component that can be incorporated into the pre-existing framework.

5.5.2 New component

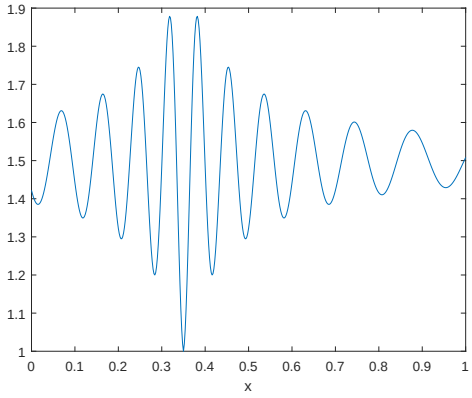
The new component is required to not be easily matched with a simple function.

$$\begin{aligned} s\_signal(x, A, B, C, D, E, F) &= D + F + \left(\frac{G}{1 + E|x(i) - A|} \right) \\ G &= D \sin\left(-\frac{\pi}{2} - \frac{C|x(i) - A|}{B + |x(i) - A|} \right) \end{aligned} \tag{5.22}$$

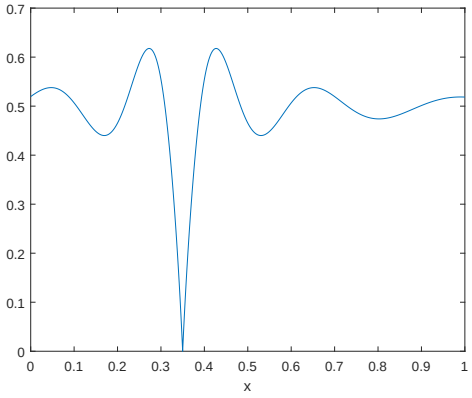
- A - Target output value - Input for which smallest output will be obtained
- B - Frequency near target - Affects minimum frequency near the target (must be > 0)
- C - Rate of frequency decrease - Affects the rate at which frequency decreases as you move away from the target (smaller values lead to a faster decrease in frequency), also effects maximum amplitude (larger values give a larger maximum amplitude)
- D - Scaling of the output/ steady state value - This value represents the the value which the sign waves deteriorates towards as you move away from the target location before offset is applied.
- E - Rate of amplitude deterioration - Determines how fast the amplitude will deteriorate towards the steady state.
- F - Vertical offset - Shifts the output vertically by the specified amount.

Two possible realisations of the component are made in Figure 5.7. These are displayed to demonstrate how the difficulty of the problem can be manipulated.

The plots present in Figure 5.8 show how the output of the component has the possibility of producing similar results for a region even though the parameters used can be quite different. This shows that possessing points within certain regions when carrying out calibration does not guarantee that the whole search space is calibrated well.

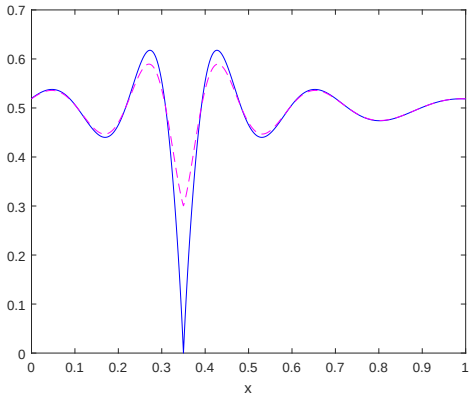


(a) Example output of component, $s_{signal}(x, 0.35, 1, 100, 0.5, 10, 1)$

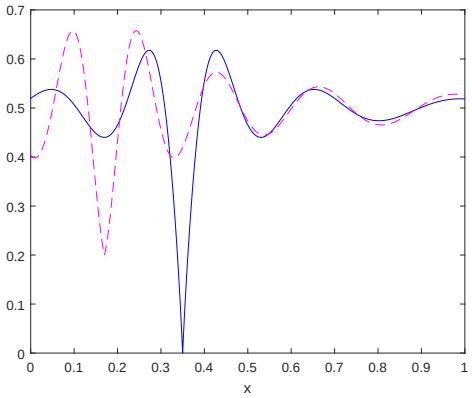


(b) Example output of component, $s_{signal}(x, 0.35, 1, 40, 0.5, 40, 0)$

Figure 5.7: Two possible realizations of the new s_{signal} component produced with fixed parameters and a varying input



(a) Output of component with $s_{signal}(x, 0.35, 1, 40, 0.5, 40, 0)$ (solid line) and $s_{signal}(x, 0.35, 1, 40, 0.2, 40, 0.3)$ (dashed)



(b) Output of component with $s_{signal}(x, 0.35, 1, 40, 0.5, 40, 0)$ (solid line) and $s_{signal}(x, 0.17, 1.4, 60, 0.3, 12, 0.2)$ (dashed)

Figure 5.8: Plots of two possible realizations of the function

It is possible to take the different parameters on the component to be the system parameters that need to be obtained in order to calibrate the system. Limitations will need to be applied on a couple of the parameters to ensure that they stay within a reasonable range and prevent them going negative in some cases.

In order for this new component to be used within the WFG framework it is necessary to ensure that the output is limited to fall within the range of 0 to 1.

5.5.3 New WFG function

In order to fulfil our objective it is necessary to use the the Non-separable component. The design that was decided upon looks similar to the WFG4 Function only with the multi modal (*s_multi*) component being replaced with our new *s_signal* component.

$$
\begin{array}{ll}
\text{Shape} & h_{m=1:M}=concave_m \\
\text{t1} & t^1_{i=1:n}=s\_signal(y_i,\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) \\
\text{t2} & t^2_{i=1:M-1}=r\_sum(\{y_{(i-1)k/(m-1)+1},...,y_{ik/M-1)}\},\{1,...,1\}) \\
 & t^2_M=r\_sum(\{y_{k+1},...,y_n\},\{1,...,1\})
\end{array}
$$

It is assumed that when modelling a more simple function would be produced than that which is considered the 'true' function. Hence, including the model error the function becomes,

$$
\begin{array}{ll}
\text{Shape} & h_{m=1:M}=linear_m \\
\text{t1} & t^1_{i=1:n}=s\_signal(y_i,\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) \\
\text{t2} & t^2_{i=1:M-1}=r\_sum(\{y_{(i-1)k/(m-1)+1},...,y_{ik/M-1)}\},\{1,...,1\}) \\
 & t^2_M=r\_sum(\{y_{k+1},...,y_n\},\{1,...,1\})
\end{array}
$$

This function is scalable and possesses both position (k) and distance (l) inputs, which can be expressed as,

$$
\begin{aligned}
Inputs &= [x_1\ x_2\ ...\ x_n] \\
n &= k+l
\end{aligned}
\tag{5.23}
$$

and the model parameters are,

$$
Parameters = [\theta_1\ \theta_2\ ...\ \theta_6] = [A\ B\ ...\ F] \tag{5.24}
$$

The Pareto optimal front will be achieved when all of the distance input values are set to the value of A. The range of all of the inputs is between $[0\ 2i]$, with i representing the current input being looked at.

Depending on how difficult you want the problem to be it would be possible to

use all or just a subset of *s_signal*'s parameters. Initially the problem is looked at with only A, B and C being used as parameters. This allows for both the shifting of the optimal of the signal and control over the frequency of the signal. Not allowing D and F to change has the benefit of helping to ensure the output of the component stays within the range of 0 to 1.

5.5.4 WFG4s

The final version of new problem, referred to as WFG4s, which was settled upon and used for testing in Chapter 7 uses the structure of the new test problem with an altered version of the *s_signal* component. The two changes made to the component was first to limit the parameters being actively varied to just θ_1 and θ_2. The second change was to use $(x(i) - A)^2$ instead of $abs(x(i) - A)$ which makes the trough wider. This resulted with the simplified version of the component being,

$$\begin{aligned} s\_signal(x, A, B) &= 0.5 + 0 + \left(\frac{G}{1 + 40(x(i) - A)^2} \right) \\ G &= 0.5 \sin \left(-\frac{\pi}{2} - \frac{40(x(i) - A)^2}{B + (x(i) - A)^2} \right) \end{aligned} \tag{5.25}$$

5.6 Conclusion

This chapter presents information necessary for carrying out benchmarking of a combined model calibration and optimization problem. It begins by presenting a selection of popular performance indicators include hypervolume and Inverted generational distance.

The chapter then goes on to present a methodology by which optimization test problems can be taken and adapted to the combined problem. This process is applied to a selection of popular test problems, both single and multi-objective, from within the optimization literature.

After examining the test problems produced it is determined that it would be necessary to develop a new function to accommodate specific properties. The main property that was lacking was for the problem not to be easily approximated via the use of a simple surrogate model. Rather than developing a new problem from scratch it was decided to develop a new component for the WFG framework.

Now that a suite of test problems is available the next step is to implement possible solutions to this joint problem. The following chapter looks at methods that can be implemented within a medium sized budget of 5000 function evaluations.

Chapter 6

Architectures for 5000 evaluations

6.1 Introduction

This chapter investigates methods for solving the combined problem of model calibration and optimization. This is done with the aim of improving the overall performance of the combined stages without the use of additional function evaluations.

The new adaptive method looks at using a combination of Bayesian calibration, implemented using MCMC, along with MOEA/D and is compared to the classical series approach. A high-level schematic of each process is presented along with details about how this specific implementation was carried out. Additional information is included, covering a discussion on how robust optimization could be achieved as well as examining a possible process for correcting model error.

There are two performance metrics by which the results will be assessed. These are the hypervolume indicator along with the absolute error between the true and apparent hypervolumes. The division of the evaluation budget is presented for both the classical and alternating approach.

Results from running both the classical and alternating methods with 21 repetitions for three different test functions are presented. The selected test functions are the DTLZ1$_\theta$, ZDT1$_\theta$ and WFG2$_\theta$ test problems. All three of these are multi-objective problems that have been adapted from within the optimization literature to include parameters that need to be determined along with the control inputs. These adapted functions can be seen in Section 5.4.4.

The chapter begins by laying out the implementation of the methods in Section 6.2. This is followed by a presentation of results and discussion in Section 6.3. The final Section 6.4 gives the conclusion that have been drawn.

6.2 Implementation

Now that the framework for laying out the combined problem and the methods for benchmarking them have been discussed the next step is to investigate the effectiveness of different solvers. As mentioned previously in Section 4.3 there are three main methods by which it would be possible to go about solving the combined calibration and optimization problem. These three methods are the classical method of performing calibration followed by optimization (covered in Section 6.2.2 and 6.2.3), Robust optimization (discussed in Section 6.2.4) and an alternating approach (explained in Section 6.2.5). Before looking at the implementation information for each of these, schematics for the classical and alternating approaches are laid out (Section 6.2.1).

6.2.1 Schematics

The components present within both schematics are set up so that their implementations are as similar as possible. The main differences between the two implementations are how the evaluation budget has been divided and what information is available at any given point. The main components are represented as boxes with the movement of information shown as lines. The notation used here is inline with that presented in Chapter 4.

Classical approach

The first of the two implementations to be considered is that of the classical approach within which the model calibration is carried out a single time before optimization is performed. In this set up once the calibration has been performed and a parameter set, or distribution, has been obtained it is never updated and is assumed to provide a reasonable representation of the system.

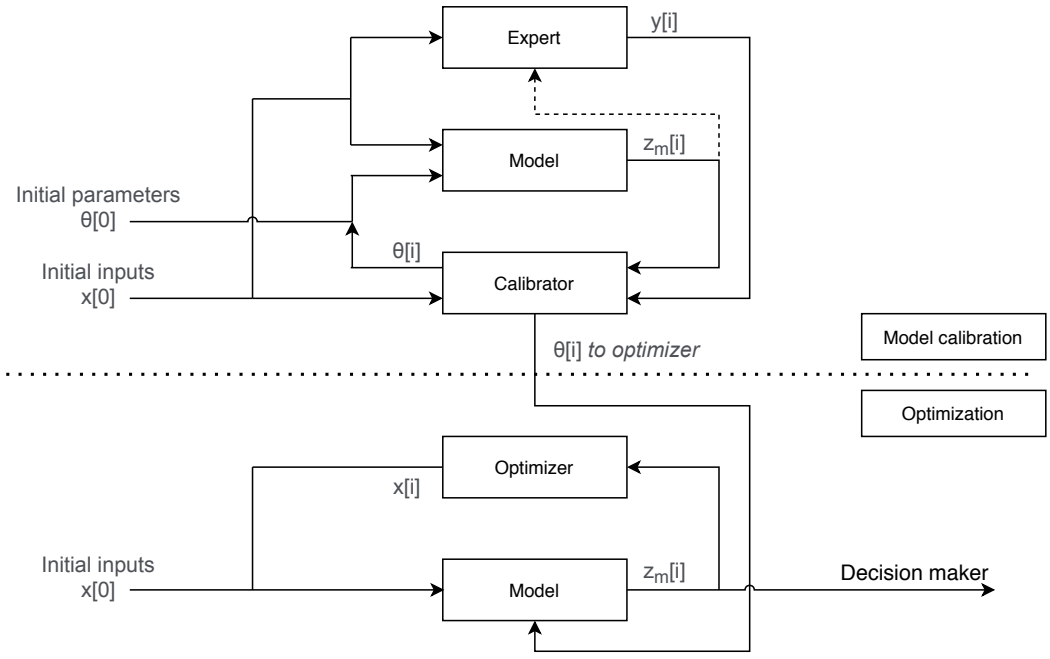


Figure 6.1: Schematic of the classical aproach for tackling model calibration and optimization

Figure 6.1 presents a high level schematic of the two processes of model calibration and optimization. Within model calibration there are three main components: calibrator, model and expert. The expert component represents an entity that knows how the system should perform for a given set of control inputs. In this study the expert is represented by a test function for which the predefined 'true' parameter set is used. The model takes both a set of control inputs and the current estimated values of the model parameters and returns an output. This work considers both cases where the structure of the model exactly matches that of the expert function and when there is a modelling error present. The modelling error is incorporated as an alteration in the test function as depicted in Section 5.4.4. The calibrator represents the combination of Bayesian calibration and MCMC. It produces a set of potential parameters using an initial set of control inputs, along with corresponding true system outputs and the model output for a previous assumption of the parameters.

The second half of the schematic shows the optimization process (see Figure 6.1). There are two main components: the model, which performs the same function as the one present within model calibration, and the optimizer. The optimizer runs a version of MOEA/D and determines the new sets of control inputs believed to have superior performance.

Once the model evaluation budget allocated for calibration has been fully utilised, the calibrator passes the selected parameter set to the optimization process. The parameter set is selected based on the maximum likelihood. When the optimization stage has expended its evaluation budget, the results are passed onto the decision maker for consideration.

Alternating approach

The second approach presented here is the alternating approach which considers moving back and forth between the stages of model calibration and optimization in order to make better use the of available function evaluation budget.

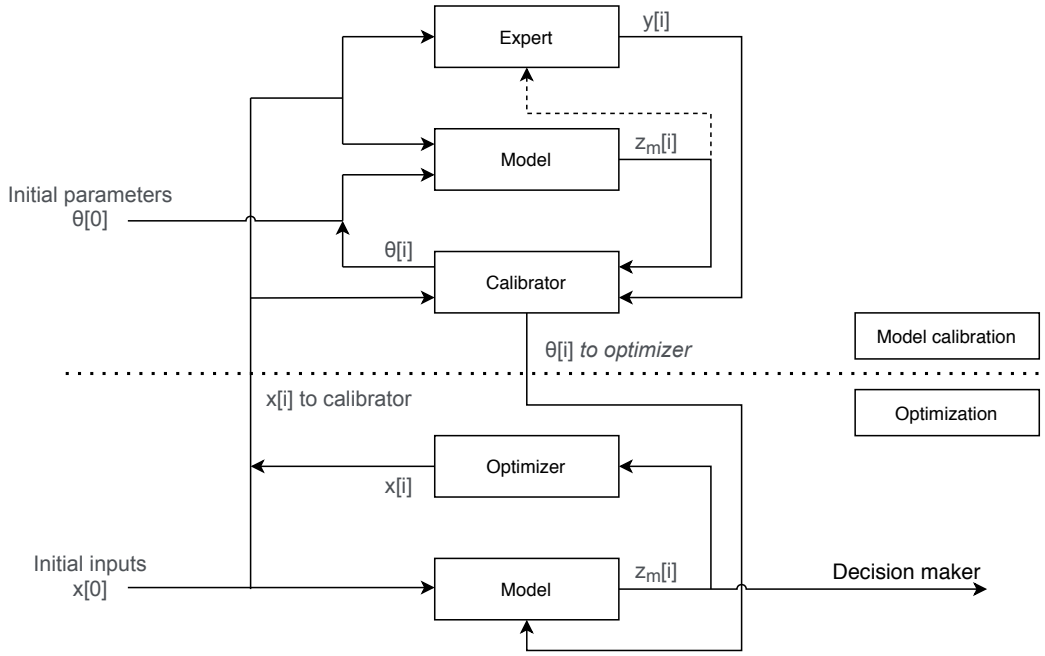


Figure 6.2: Schematic of the new alternating aproach for tackling model calibration and optimization

As can be seen in the schematic presented in Figure 6.2 the alternating method has much the same structure as that of the classical approach. The main difference between the two setups is that there is now a connection which can pass new control inputs back to the model calibration. This connection is what allows for an iterative method to occur, with the aim of ensuring that the information gained focuses on the areas of interest. The switching condition between the two stages for this work is set to be when a predefined portion of the available evaluation budget has been used. The next section covers the implementation information for the main components of the calibrator and optimizer.

6.2.2 Model calibration

Model calibration is implemented using a form of Bayesian calibration (Section 2.2). Markov Chain Mote Carlo (MCMC) sampling, (Section 2.2.2) was used to sample the distribution produced and determine a set of the parameters based of the maximum likelihood. In specific Gibbs sampling was implemented using the

Metropolis Hastings acceptance criteria for new points. This allows for each of the parameters to be developed as well as allowing the possibility of backwards movement within the output space. The standard deviations used for sampling the parameters of new points within the chain are determined at the start using the range of each of the parameters. The following Implementation decisions were made for model calibration:

- The MCMC starting point was randomly selected from within the multidimensional parameter space
- A normal distribution is used to determine the next point in the MCMC chain, with the mean being taken as the previous parameters and the standard deviation being a predefined value, sigma. The value of sigma used varies depending on the problem being looked at and is defined as a proportion of the parameter range. Due to this it is necessary to calculate the value of sigma before beginning.
- If a newly determined set of parameters lies outside of the acceptable range they are discarded and a new set is produced. In the case that none of the samples are viable within a reasonable number of samples an error is given to the user.
- The likelihood used for MCMC was calculated using the log-likelihood function,

$$p(D|\boldsymbol{\theta}) = -\frac{N}{2}\log(2\pi) - \frac{N}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1:N}(y_i - f(\boldsymbol{\theta}))^2 \tag{6.1}$$

  where y and $f(\theta)$ follow the notation from Chapter 4 and are assumed to have been samples using the same control input sets. N defines the number of control input sets that have been sampled and σ is the standard deviation of the noise from the points that have been evaluated on the physical system. In this work σ is treated as a known value.
- The acceptance criteria and the calculation for the acceptance ratio used in this work are the general ones defined for the Metropolis Hastings methodology
- An extension to this method would be the use of a Least squares estimate. This is an optional stage that can be performed to correct for the modelling error. See section 6.2.6.

6.2.3 Optimization

In this chapter we once again use the Multi Objective Algorithm Based on decomposition (MOEA/D) algorithm used in Chapter 3. As discussed previously MOEA/D is a seminal algorithm within the field and its wide use makes it a suitable benchmark for comparison to other methods. The key implementation choices included:

- The use of ten reference directions that are set to be evenly spread over the output space. While in some cases this can be sub-optimal, when no knowledge is present it helps to ensure that a large proportion of the output space is explored. It should be noted that no specific scheme is used to assign points to reference directions and instead they are assigned at random. This could prove to be suboptimal as time could be waisted traversing the output space.
- Neighbourhoods as mentioned before are a new aspect introduced within the MOEA/D algorithm. For this work a neighbourhood size of 3 was used due to the limited budget meaning only 10 reference directions were present.
- The distribution index used for reproduction is taken to be 20 within the implementation presented.
- The scalarisation function selected for implementation within the optimizer is the Tchebycheff function. This function was chosen over other possible alternatives such as weighted sum due to its ability of providing guidance as the point evolves and draws the point towards its reference direction. This feature proves to be useful as it ensures that a spread of points across the Pareto front is achieved.
- When determining the outputs for use within the MOEA/D algorithm, which will be used to decide if a new point should be kept, the normalised values are used. The DLTZ1$_\theta$ uses a normalisation of [1,1], ZDT1$_\theta$ uses [1,5] and WFG2$_\theta$ uses [2,4].

6.2.4 Robust optimization

One of the possible solutions was to implement robust optimization instead of using some form of Model calibration to reduce the modelling error. The implementation of robust optimization can be achieved through replacing the output value of the system with some form of performance indicator. Possible performance indicators which could have been used include the Maximin, Mean and 90^{th} percentile

When thinking of implementing a robust strategy there lies a potential issue due to the number of evaluations that are required. In the case of all three of the indicators mentioned, the usual method for implementation is to produce a population of points based on each of the new output points that have been discovered and then calculate the indicator from that. Here, it is impractical as the number of function evaluations being used is limited.

There are methods to circumvent this through defining a distribution, although this has not currently been explored. Such methods can include the use of implicit averaging and estimation of the robustness indicator such as is seen in the paper by Duro et al. (2019). This is one area which should be investigated within future work.

6.2.5 Alternating

The third option proposed was to use a scheme that would alternate between the stages of model calibration and optimization in order to try and ensure that the evaluations that are used provide the greatest benefits possible. In order to do this, it is necessary to develop a criterion for when to switch between the two stages. These could include switching conditions such as,

- Switching after a predetermined number of evaluations have been used,
- When the improvement obtained from an evaluation, or set of evaluations, falls below a certain amount,
- After a random number of evaluations,
- If the optimizer performance starts to get worse
- If the model calibration parameter values have stabilized.

For testing it was decided that the simple option of switching after a predetermined number of evaluations were used would be implemented. The next aspect to be considered was when the true function evaluations, expert points, would be used. In the classical approach they are all used to create an initial population before model calibration is started. For the alternating approach the available true evaluations are split between being used at the start and after each time the optimization budget has been used up before the next iteration of calibration is carried out. There are four different options for going about selecting these new expert points which have been examined. These are,

- Do not add any new points to the expert population,
- Asses all undominated points found by the optimizer,

- Asses a subset of the undominated points found (if only a limited budget is available)
- Assess points gained from some form of sampling technique e.g Latin hypercube.

In order to assess what impact the types of selected points would have, if used within the alternating setup, an experiment was run. The alternating approach was implemented for the DTLZ1$_\theta$ function with five alternations. The only difference in the setup was the method by which new points were selected and added to the expert population. The resulting Inverse Generational Distance (IGD), detailed in Section 5.2.3, obtained from this can be seen in Figure 6.3.

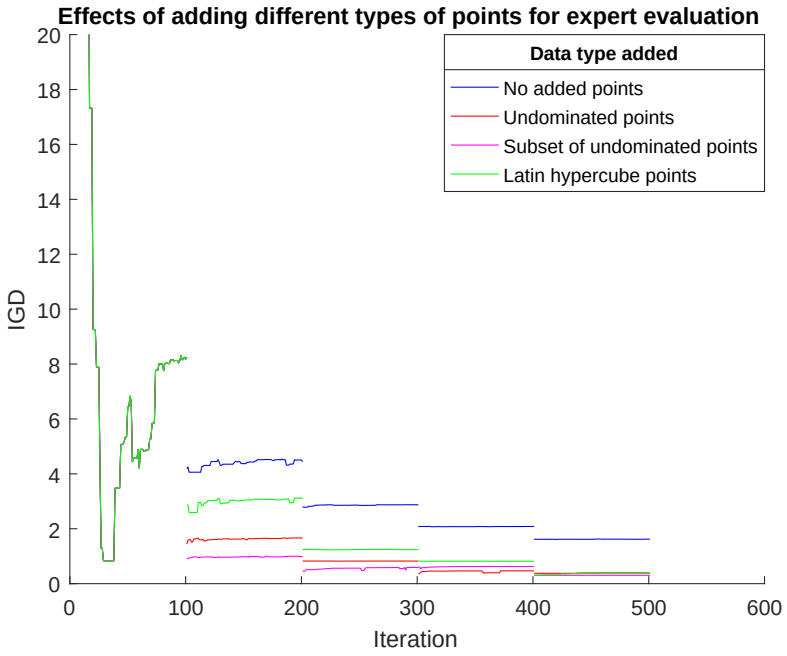


Figure 6.3: Comparison of the effects of different methods by which new points are added to the expert population during an implementation of the alternating method. Results are shown as a comparison between the achieved IGD and the number of optimization iterations that have been carried out. A greater improvement, smaller IGD values, can be identified when more information is made available to the method as well as when the information more directly relates to the area of interest. Once a sufficient quantity of information is present the impact of the chosen selection methods is reduced.

The first thing to be noted from Figure 6.3 is that the progress of the optimizer over the first 100 optimization evaluations is the same when using all four methods as at this point there is no difference present within the implementations. After the first loop of the code is completed (both calibration and optimization) the impact

of the methods can be observed. In the case when no additional points are added to the expert population the performance is significantly worse than that of the other proposed methods regardless of the type or amount of information added.

With the exception of the first 100 optimization evaluations, during each subsequent 100 optimization evaluations little to no progress can be observed from the optimizer. This is due to it having already achieved what it believes to be the optimum Pareto front. This excessive number of evaluations has been permitted as the impact of the selection method was of interest rather than how the optimizer would affect it. The oddity seen during the first 100 evaluations where the performance appears to improve and then become worse again is due to the obtained points passing the true front. Similar performance to this can be seen in Section 6.3.1.

The large leaps in performance are observed as more knowledge of the system is obtained and hence better calibration is achieved. As the improved parameters are found it becomes possible to obtain points that lie closer to the true Pareto front and hence superior IGD value are obtained. One aspect to take note of is the case when the additional points were chosen using Latin hypercube sampling. In this case while an improvement is seen, it is initially not as effective as when information more closely relevant to the problem was added. After enough points were added however the impact of this is greatly reduced, it is assumed that this reduced impact is due to enough relevant points being determined near the area of interest by this time. Both methods in which undominated points were selected for addition to the expert population are shown to be effective. It is unclear why a subset of the undominated points outperformed the full set. One possible explanation is that the subset prioritised selecting undominated points and so the calibration was more focused on the region of interest (the area closer to the Pareto front).

6.2.6 Correcting model error

The problem of the discrepancy between the true system and model, caused by the model error, needs to be considered. The plots in section 5.4.4 demonstrates that even if the optimal inputs and parameters are used there will still be an error due to the structural error present. In most of the problems this took the form of the Pareto front possessing an altered shape. In order to combat this error, it is first necessary to determine what portion of the error present is due to parameters and what portion is due to inaccuracy in the model structure. During implementation this is hard (if not unfeasible) to do, so instead of this once calibration is performed all remaining error is assumed to be a result of inaccuracies in the model.

The exact values of the model error at each of the true evaluations performed on the system can be calculated without the need for further evaluations. Using this limited set of data, it is necessary to be able to correct for the model error. One possible way of performing this correction would be to implement least squares estimation (LSE) using the available output data and points with known model error. Performing LSE it is possible to fit a line comparing model error against each of the objectives and use that to estimate the error for any given output. This is one way to produce a model of the model error. There is one major issue with this however, which is that there is no certainty that there will be a correlation between the different values of the output. Another issues is that the aleatory uncertainty (ϕ) might be included, this should only really be an issue though if the model error is of similar size or smaller than ϕ. An example in which least squares estimation was applied to find a correction term as described can be seen in Figures 6.4 and 6.5.

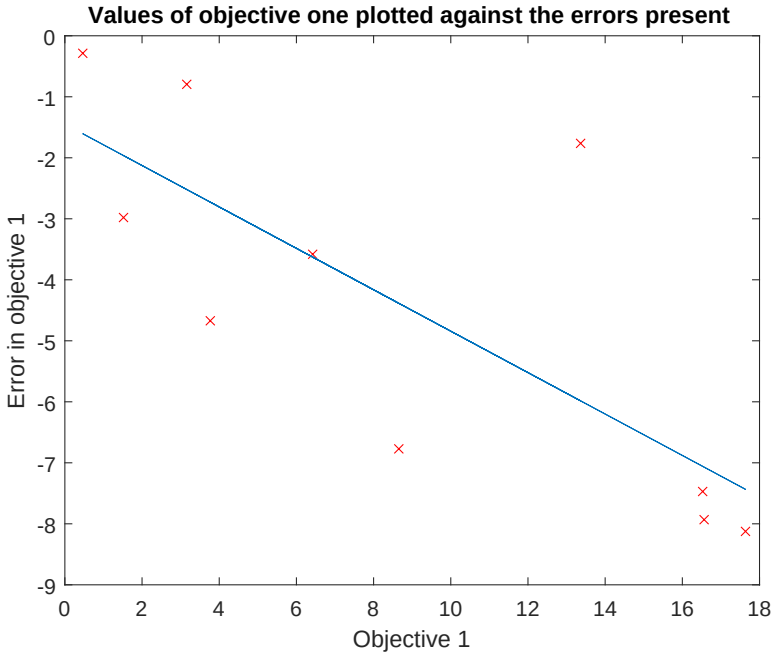


Figure 6.4: The least squares estimation of the error between the true system output and the modelled output for the DTLZ1$_\theta$ function.

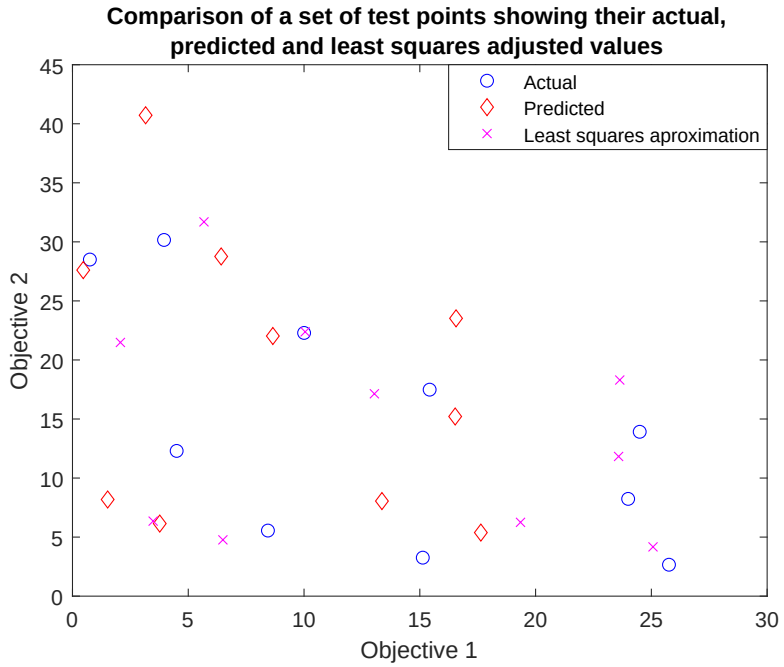


Figure 6.5: The effects of applying the least squares estimation results as a corrective term to the model outputs after calibration for the DTLZ1$_\theta$ function.

In Figure 6.4 the LSE for the error in objective 1 is presented and although the estimate appears to fit there is still a large difference between the true and predicted values of the error for some of the points. The effects of applying this LSE, along with a corresponding one for objective 2, to the predicted outputs can be seen in Figure 6.5. Doing this does appear to aid in improving the model estimate of the output. In most cases the output, after being adjusted using the LSE, lies closer to the actual objective output than the value obtained through the calibrated model by itself.

The traditional method for applying the LSE to a problem like this would be to use it to model the effects of changing the inputs on the output values. Ideally when performing this correction it would be ideal to use points in which only a single input is varied at a time. Unfortunately, with the current setup this information is not currently available. While obtaining these points would be plausible, it would require further function evaluations to be undertaken with the consequential increase in cost. This in turn would reduce the available budget present during the calibration or optimization stages. Due to this it is necessary to obtain points for which multiple inputs are being varied. The issue of such points is that it is harder to identify the impact that the variation of each of the inputs is having, this issue is confounded by the fact that only a limited number of points is available in the first place due to the limited evaluation budget.

The results of applying LSE examining the effects of varying inputs on the outputs can be seen in Figure 6.6. During the implementation only five points were used for training, with an additional ten to demonstrate what effect it has. The five training points can be identified within the figure as those which have their LSE approximation almost perfectly matching the true points. While this implementation of the LSE does appear to perform well there are a couple of the LSE approximations (identified at the far right hand side of the plot) which still lie far from the actual objective values for the given sets of inputs. In addition, in some cases the LSE can perform worse than the predicted values.

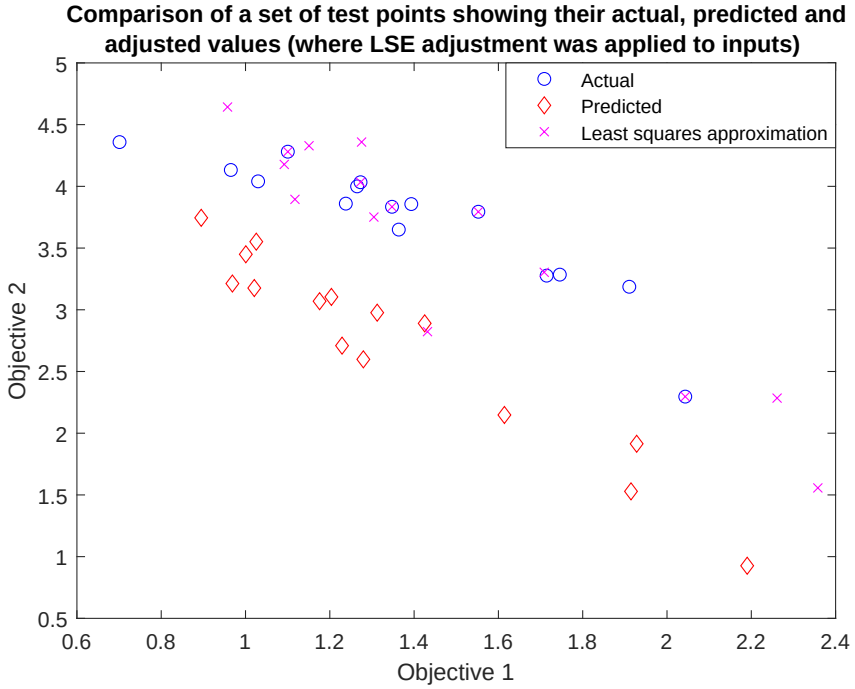


Figure 6.6: The effects of applying the least squares estimation results as a input correction term for the model outputs after calibration on the WFG4$_\theta$ function

6.2.7 Performance measure

As discussed earlier, in Chapter 5 there are many performance metrics which could be used for this work with the main two that have being considered being the inverse generational distance (IGD) (Ishibuchi et al., 2016) and hypervolume (Bader, 2010). When considering the use of IGD for this work there is a problem as the choice of what to take for the Pareto front can potentially change depending if the model or true system are being examined. Looking to the future there is also the issue that for real world problems, knowledge of the true front would not be available, so while the use of IGD might be beneficial for visualisation in some

cases, it is not necessarily practical here.

Hypervolume is the second performance measure considered, usually for testing purposes IGD would be more suitable as it is necessary to select a reference point for hypervolume which has the possibility of introducing bias on the different objectives. One possible option for selecting the reference point would be to take the nadir point (either as it is or with some gain applied) as this would ensure that it would both fully include the Pareto front as well as help to prevent bias. For this work this is not practical as it is desired to assess the progress of the evolving Pareto front throughout the optimization process. If the nadir point was used a large proportion of the resulting hypervolumes would be shown as zero due to the current best undominated set of points from the population lying outside of the reference point for some of the test problems. Due to this the reference point selected for the hypervolume calculation was found by selecting the worst case for each of the objectives from a combined population. This ensures that all points found will produce a positive hypervolume value. In addition to this the hypervolumes presented are shown as a percentage of the optimal hypervolume with 100% representing perfect knowledge of the Pareto front.

Another performance indicator that is looked at for this work is the absolute error between the true and apparent hypervolume. This works as an indication of the amount of error present both from the model and parameter error. The error of the hypervolume is of specific interest as it embodies the error within the points present in the undominated set. These points lie close to the true Pareto front and so represents the area of the search space at which it is most desirable to have a reduced model error.

Statistical assessment

After the results from implementing different possible methods have been obtained and performance measures have been calculated, the next step carried out in this study is to assess what impact the different methods have actually made. To do this permutation testing has been implemented. Permutation testing works by resampling a population numerous times in order to obtain a p-value, the probability of getting data as extreme as or more extreme than the observations for which the null hypothesis is true. Permutation testing has a few advantages including,

- Being simple to implement
- Not requiring knowledge of the distribution of the data

- Ability to analyse unbalanced designs

There are also disadvantages,

- Depending on the number of data points the resulting number of permutations required can be restrictive
- Observations need to be exchangeable under the null hypothesis

In this work it was decided to do 21 repeat runs for each of the setups. This means that if two of the setups are compared using permutation testing there would be 3.122e+23 possible permutations that should be considered. This is simply not practical to compute even with the recent advancement in computational power. In order to get around this a subset (1,000,000 permutations) of the possible permutations are selected at random for use in calculating the p-value. To carry out permutation testing the results of both setups are combined into a single population which is then permuted and broken down into two new populations. The means of these two populations are then compared to assess if the difference between them is larger than the difference between the means of the original population. This is repeated numerous times and the p-values is calculated by dividing the number of times the difference between the means was larger by the number of permutations performed.

The null hypothesis that is being used for this work is that there is no difference according to the performance indicator. The alternative hypothesis states that there is a difference. The statistical assessment considers the problem, without Bonferroni correction, as being two tailed which means that the required p value is 0.025 rather than the standard 0.05.

6.2.8 Evaluation budget

There are two types of evaluations that need to be considered when determining how the system should be run. These are the true function evaluations that come from physical experiments or expert opinions and the expensive model evaluations. In both cases, it is assumed that the budget will be limited due to factors such as expense or time constraints. In this work a slightly larger number of model evaluation than would normally be used for expensive problems has been selected.

For model runs a budget of 5000 evaluations was chosen due to it being one order of magnitude larger than the usual budget of around 500 evaluations. The true system evaluations (expert points) are limited to just 10 and are used within

the calibration. When sampling the expert points there is observation error applied. For this work the observation error is produced by a normal distribution with the mean set on the objective value and a standard deviation of either 0.1 or 0.05 depending on the problem. The schedule of the evaluations used differs between the two setups as follows.

- Classical
  - The full 10 expert points are evaluated at the start
  - Calibration uses 3000 model evaluation – Broken down into 300 MCMC steps with comparing to the 10 expert points.
  - Optimization uses 2000 model evaluations – Broken down into 200 optimization iterations with 10 reference directions being examined for each.
- Alternating
  - Half of the expert points, 5, were evaluated at the start with a further 5 being evaluated after the first iteration of optimization was complete.
  - Calibration uses 3000 model evaluations in two segments, with 1000 evaluations being used for the first calibration and 2000 for the second – In both cases 200 MCMC steps were used, although only 5 expert points were available for comparison in the first case with 10 available in the second.
  - Optimization used 2000 model evaluations split equally between the first and second optimization stage – This works out as two repetitions of 100 optimization iterations with 10 reference directions.

6.2.9 Input and parameter ranges

The configurations of the three test problems used, DTLZ1$_\theta$, ZDT1$_\theta$ and WFG2$_\theta$, is presented in Section 5.4.4. During implementation it was necessary to set up minimum and maximum values for each of the control inputs and parameters and while the limits of the control inputs was discussed before they have been restated here for easy access. The true values presented here represent the control inputs required to get optimal performance and the parameters required to remove parameter error from the model.

DTLZ1$_\theta$

	Control Input			Parameter				
	i=1	i=2:6		1	2	3	4	5
Maximum	1	1		2	1.5	15	1	0
True	-	0.5		0.5	0.5	5	-0.5	-0.5
Minimum	0	0		0	0	0	-3	-1

Table 6.1: The minimum, maximum and true values for the parameter and control input values present within the DTLZ1$_\theta$ function.

ZDT1$_\theta$

	Control Input			Parameter	
	i=1	i=2:30		1	2
Maximum	1	1		4	1
True	-	0		1	9/29
Minimum	0	0		0	0

Table 6.2: The minimum, maximum and true values for the parameter and control input values present within the ZDT1$_\theta$ function.

WFG2$_\theta$

	Control Input			Parameter	
	i=1:4	i=5:24		1	2
Maximum	2i	2i		6.5	1
True	-	0.35i		5	0.35
Minimum	0	0		4	0

Table 6.3: The minimum, maximum and true values for the parameter and control input values present within the WFG2$_\theta$ function.

6.3 Results and discussion

The results of assessing the performance of the three test problems, DTLZ1$_\theta$, ZDT1$_\theta$ and WFG2$_\theta$, is presented within this section. In each case the test problem has been used with both the classical approach of performing calibration followed by optimization (series method) and the alternating approach. The results are presented as four figures, with the first showing the final parameter values achieved by the methods. The second and third figures show the apparent and true hypervolumes achieved by the methods. The final figure depicts the absolute error

between the apparent and true final hypervolumes achieved. All of these are results are shown as boxplots and represent 21 repetitions on the methods. The results demonstrate that the performance obtained by the alternating is either an improvement or comparable to that of the classical (series) method. There are an additional two figures presented for the DTLZ1$_\theta$ function which look at the log-likelihood achieved and present a set of results within the output space.

6.3.1 DTLZ1$_\theta$ function

What was the log likelihood observed during the calibration

The first aspect to the methods to be examined is the progression of the calibrator over the course of a run. This is shown in Figure 6.7 as the log-likelihood achieved at each step of the MCMC. The four cases depicted here are when the alternating and series (classical) approach were run for both the cases when modelling error was present and absent. The difference in the number of steps between the alternating and series method is explained in Section 6.2.8. When looking at these results it is important to remember that the expert points used to calculate the likelihood differ between the two methods so performing a direct comparison of the maximum values reached can be misleading.

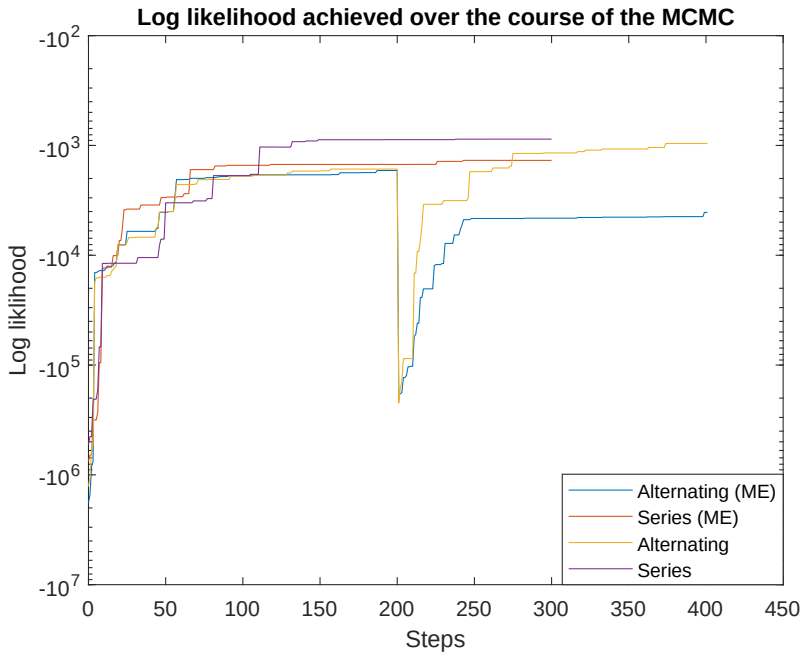


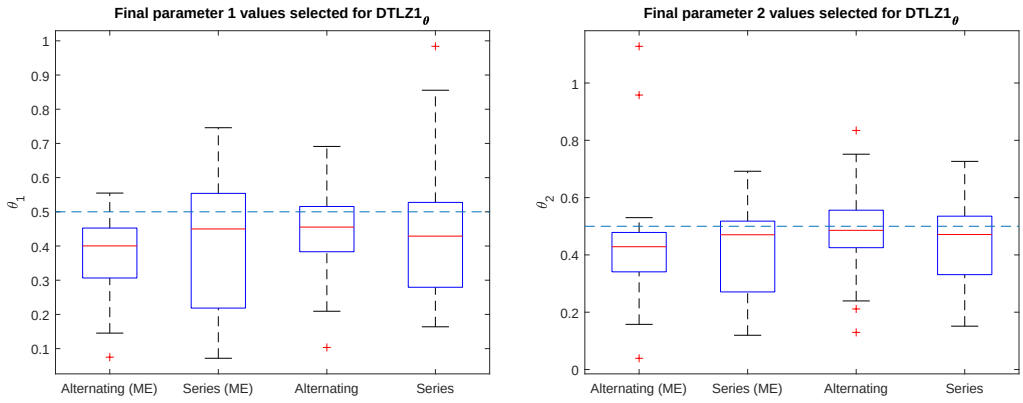
Figure 6.7: The progress of the log likelihood obtained by the MCMC over the course of the calibration. The alternating and classical (series) methods are depicted for both when modelling error (ME) is present and absent.

During the series runs, fast initial improvement is seen, as better parameter sets are discovered. By around 150 steps however, the rate of improvement is greatly reduced with little change present for both when the model error is present and absent. Looking at the alternating method a similar rate of improvement is observed during the first calibration stage. After the 200th step when the new expert points are gained by the alternating method, a drop in the log likelihood can be seen as the algorithm gains a better knowledge of the system. Once the alternating algorithm commences its second calibration stage rapid improvement is again seen. The large difference between the final obtained log likelihood for the alternating method when model error is present and absent is likely due to there not currently being a corrective term applied, to remove the modelling error.

Examination of the final parameter values obtained by the two methods when testing with the DTLZ1$_\theta$

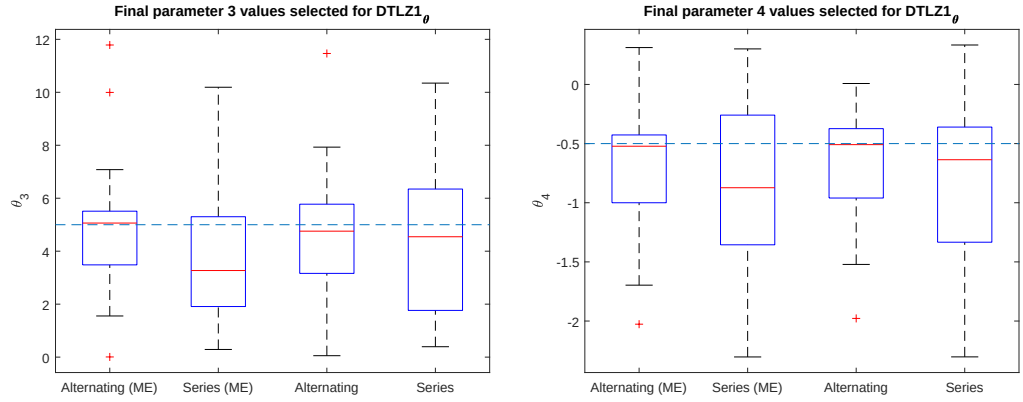
The final obtained values for the five parameters can be seen in Figure 6.8. The first thing to notice is that the calibration runs better when no model error is present. The reason why both methods struggle to identify the correct parameter values when model error is present, is that the model error changes the parameters required for an evaluated point to match the expert value for a given set of inputs. This results in values close to the true parameters producing a worse likelihood than those in a different location and so getting rejected by the MCMC.

The parameters each have a different impact on the Pareto front with some having greater importance for optimization than others. Both parameters 1 and 2 work to scale the objective values. While incorrect selection of these may affect the speed of the optimizer it is still possible to acquire the current control inputs. Parameters 3, 4 and 5 have a more direct impact on the control inputs being selected. In the case of parameters 3 and 4 it is possible to find combinations that can make it look like the problem is correctly calibrated while it is not. Due to this it is important to ensure they are well calibrated. Turning back to the figure, for these two parameters the alternating method archives a better calibration than that of the series method. It should also be noted that for all five of the parameters the final distributions achieved by the alternating method are smaller than those seen for the series method.



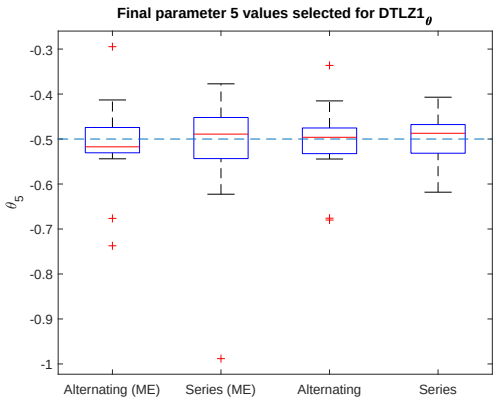
(a) Parameter 1

(b) Parameter 2



(c) Parameter 3

(d) Parameter 4



(e) Parameter 5

Figure 6.8: The final parameter values achieved by the alternating and classical (series) methodologies for both the cases when modelling error (ME) is present and absent while examining the DTLZ1$_\theta$ function.

Comparison of the apparent hypervolumes achieved by the classical and alternating methods

The apparent hypervolume examined within these results is obtained from the resulting points for which the model was evaluated using the current best knowledge of the parameters. In Figure 6.9 there are many cases in which the achieved hypervolume exceeds 100%. These scenarios can occur due to the incorrect calibration of the model. The alternating approach appears to perform more consistently well when compared to the series method, both when model error is present as well as when it is not. It should be noted that there are still outlying cases in which the alternating method performs badly. Another interesting thing to note is that the presence of model error seems to have had a larger impact on the series method causing the results to be much more spread out. The fact that the alternating method tends not to overestimate the value of the hypervolume with the 75th percentile of runs coming in at under 100% indicates it is likely to be a better representation of the true performance. In comparison the classical method regularly overestimates the hypervolume, especially when there is model error present.

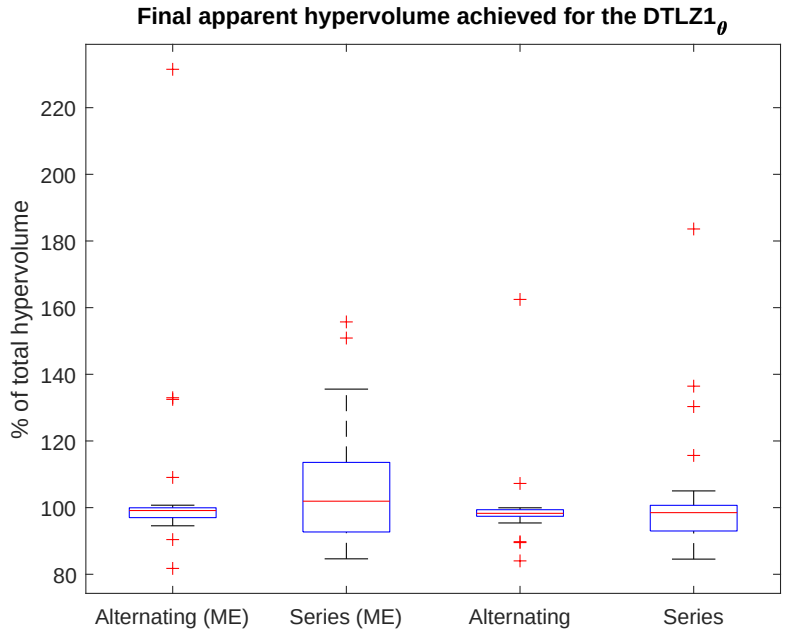


Figure 6.9: Comparison of the final apparent hypervolume achieved by the alternating and classical (series) methodologies, for both the cases when modelling error (ME) is present and absent. The y value shows the percentage of the optimal hypervolume which would be achieved using a reference point takes as the worst case for each objective in combination with the true front.

Comparison of the true hypervolumes achieved by the classical and alternating methods

The results presented in Figure 6.10 show the same points as were seen for the apparent hypervolume in Figure 6.9 but they are now being evaluated on the true system. The performance of the alternating and classical methods is very similar when modelling error is present. When no modelling error is present, the series method appears to outperform the alternating method. Testing to see if it is truly statistically better a p value of 0.7974 was obtained showing that the series method does not provide a significant improvement. From this it can be deduced that even if the determined parameters are suboptimal it is still possible to determine good inputs.

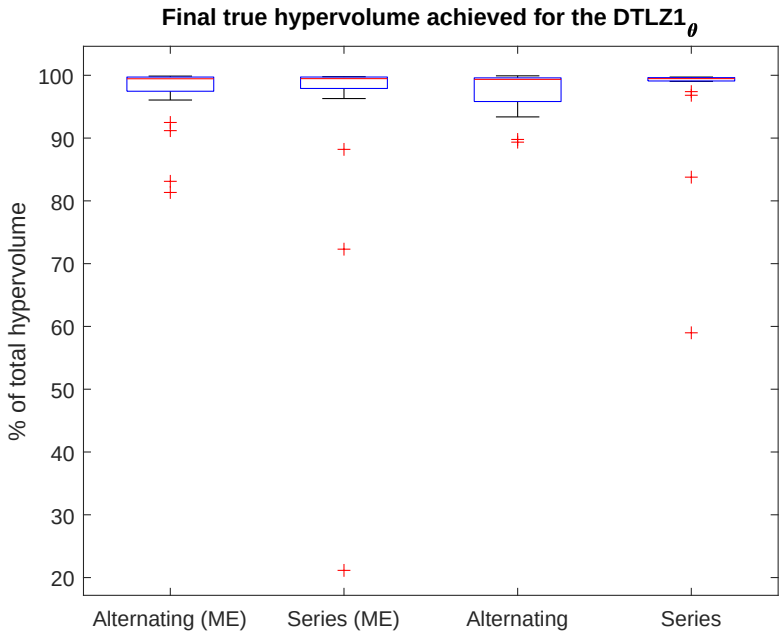


Figure 6.10: Comparison of the final true hypervolume achieved by the alternating and classical (series) methodologies, for both the cases when modelling error (ME) is present and absent. The y value shows the percentage of the optimal hypervolume which would be achieved using a reference point taken as the worst case for each objective in combination with the true front.

Comparison of the absolute error between the apparent and true final hypervolume for the classical and alternating methods

Comparing the absolute difference between the apparent and true hypervolume achieved is useful as it gives an indication of how much the user would be able

to trust the suggestion from the model. It is important to remember that the decision maker would realistically not possess the true system outputs unless they have specifically put aside some of the expert evaluations which would reduce the amount available during the implementation. The absolute difference observed is a lot smaller for the alternating methods (Figure 6.11) due to the better calibration. For the series method, when model error is present, the majority of results presented by the solver would be misleading until evaluated on the true system.

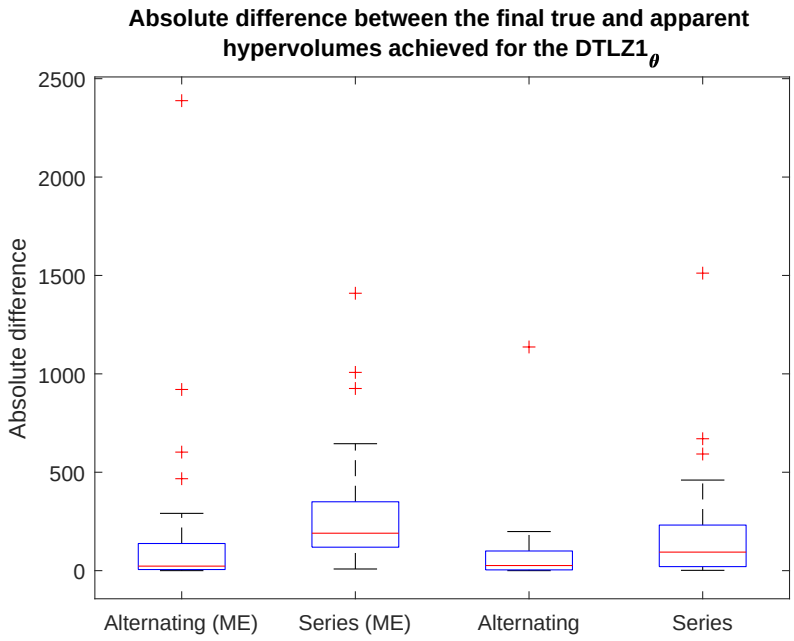


Figure 6.11: Comparison of the absolute difference between the apparent and true final hypervolumes achieved by the alternating and classical (series) methodologies both for the cases when modelling error (ME) is present and absent.

A clearer look at the true and apparent Pareto fronts for the alternating and series methods when modelling error is present

In order to get a clear understanding of how these results actually translate to the objective space the final output populations achieved by the alternating and series methods when model error was present are shown in Figure 6.12. Specifically, the runs presented are the medium performing alternating run and the series run possessing the same seed. This means that apart from the expert population both setups possessed the same initial information. The true output populations of the methods can be seen to perform equivalently. Looking at the apparent outputs achieved, the outputs of the series method are projected further from the front

than that of the alternating method. It is also interesting that the series method overestimates the fronts location while the alternating methods underestimates it.

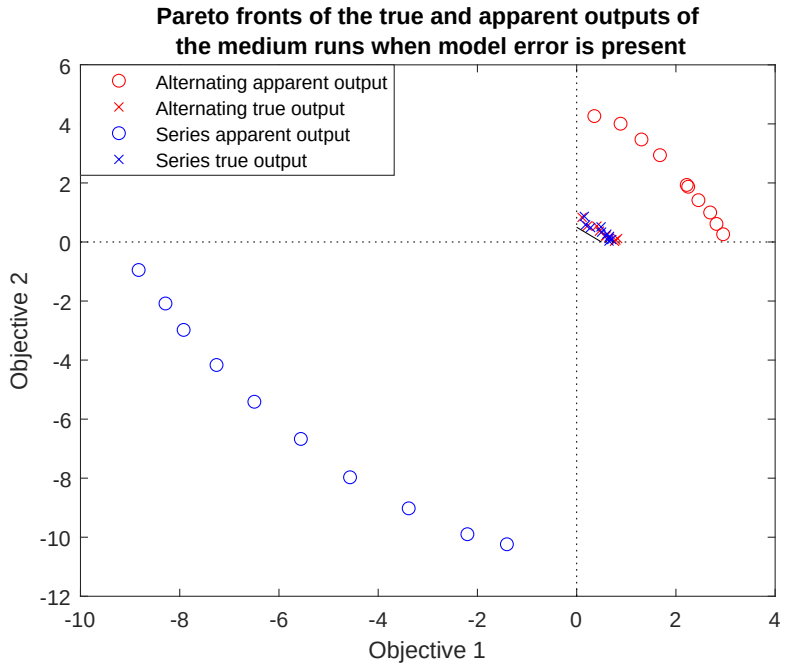
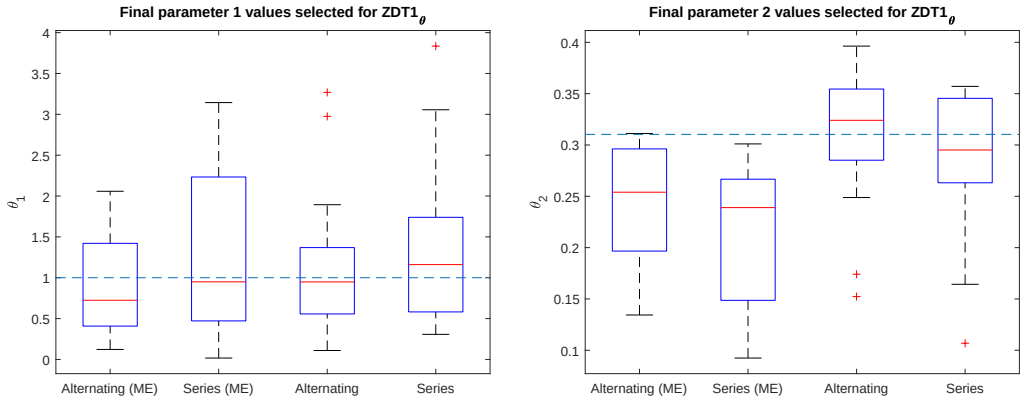


Figure 6.12: The true and apparent final populations for the alternating and series methods when modelling error is present. The populations presented represent the medium performing alternating run along with its corresponding series run.

6.3.2 ZDT1$_\theta$ function

Examination of the final parameter values obtained by the two methods when testing with the ZDT1$_\theta$

The final parameters that where achieved by the alternating and classical methods are seen within Figure 6.13 and show similar properties to those observed for the DTLZ1$_\theta$ function. Once again, the spread of the parameters is smaller the alternating method, both when model error is present and absent. It is evident that when model error is present the methods have struggled to identify the true value for parameter 2. In the case of the series method when model error was present the spread of found points did not even manage to encapsulate the true value. In contrast to this, for parameter 1, the series method successfully achieved a median value much closer to the true value than the alternating method when model error was present.



(a) Parameter 1 (b) Parameter 2

Figure 6.13: The final parameter values achieved by the alternating and classical (series) methodologies for both the cases when modelling error (ME) is present and absent. The alternating methods show a reduced spread of final value over what is seen for the classical method. Better calibration is observed when there is no model error present within the system.

Comparison of the final apparent hypervolumes achieved by the classical and alternating methods when testing with the ZDT1$_\theta$ function

Both the alternating and classical method can be seen to struggle to achieve a good final apparent hypervolume within the allotted evaluation budget, shown in Figure 6.14. This is accredited to the increased number of control inputs that are required to be calibrated within the ZDT1$_\theta$ function. While the 2000 evaluations available for the optimization stage are considered a lot from the perspective of solving expensive multi-objective problems, for the current methodology it proves to be insufficient. This shows that the development of a method that would make more efficient use of the available budget is necessary. Returning to the results, the performance of both the alternating and series methods is similar with the alternating having a slightly better median performance, both when model error is present and absent. Unlike with the DTLZ1$_\theta$ function the spread of the points found is not significantly better for the alternating method, this could be due to several factors. One example of such a factor would be, as optimization has progressed so little, the points do not lie near the Pareto front. Owing to this the cases where the methodologies overestimated the location of the Pareto front due to bad calibration, as was seen for DTLZ1$_\theta$, have not occurred.

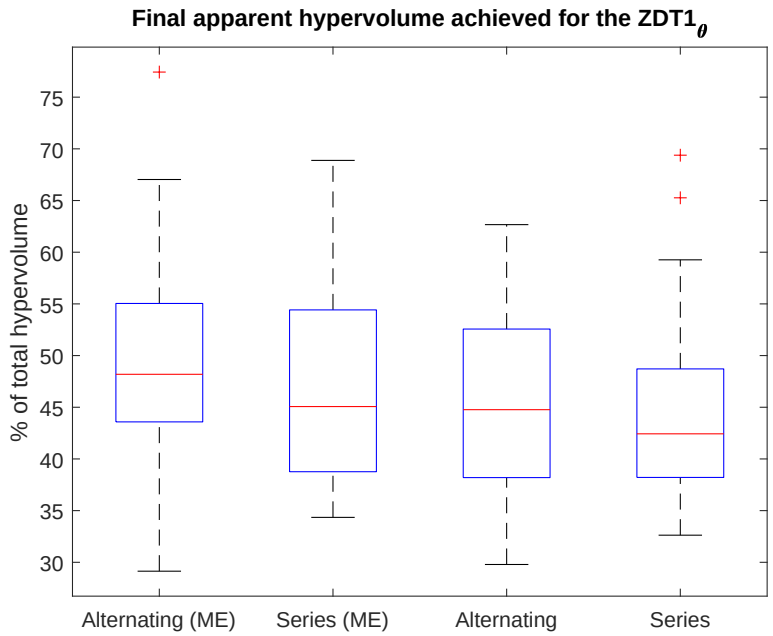


Figure 6.14: Comparison of the final apparent hypervolume achieved by the alternating and classical (series) methodologies when examining the ZDT1$_\theta$ function, for both the cases when modelling error (ME) is present and absent. The y value shows the percentage of the optimal hypervolume which would be achieved using a reference point takes as the worst case for each objective in combination with the true front. Both the alternating and classical methodologies struggled to achieve a satisfactory final apparent hypervolume.

Comparison of the final true hypervolumes achieved by the classical and alternating methods when testing with the ZDT1$_\theta$ function

The final true hypervolumes obtained by the alternating and classical methods, displayed in Figure 6.15, possess similar performance to the final apparent hypervolumes achieved. The alternating methods maintain a very slight lead when looking at the median performance, although this lead is not statistically significant. This is especially evident when comparing the medium performance of the two methodologies when modelling error is present.

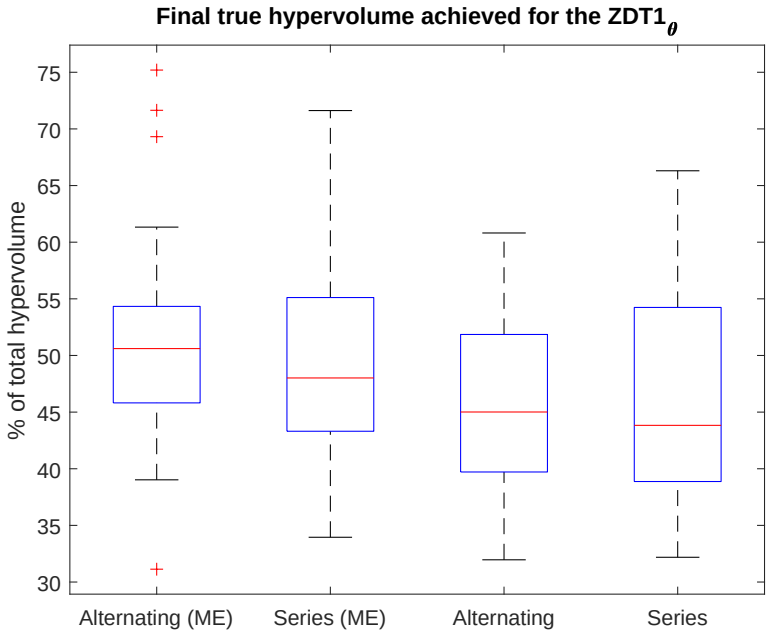


Figure 6.15: Comparison of the final true hypervolume achieved by the alternating and classical (series) methodologies when examining the ZDT1$_\theta$ function, for both the cases when modelling error (ME) is present and absent. The y value shows the percentage of the optimal hypervolume which would be achieved using a reference point takes as the worst case for each objective in combination with the true front.

Comparison of the absolute error between the apparent and true final hypervolume achieved for the ZDT1$_\theta$ function using the classical and alternating methods

From examining the absolute error between the apparent and true hypervolume, displayed in Figure 6.16, it can be seen that the alternating method manages on average to obtain smaller values. Looking back at the absolute error observed for the DTLZ1$_\theta$ function the current results match the behaviour which was previously present. This behaviour once again indicates that the results produced by the alternating method are a more reliable representation of the true system, when additional true function evaluations cannot be carried out.

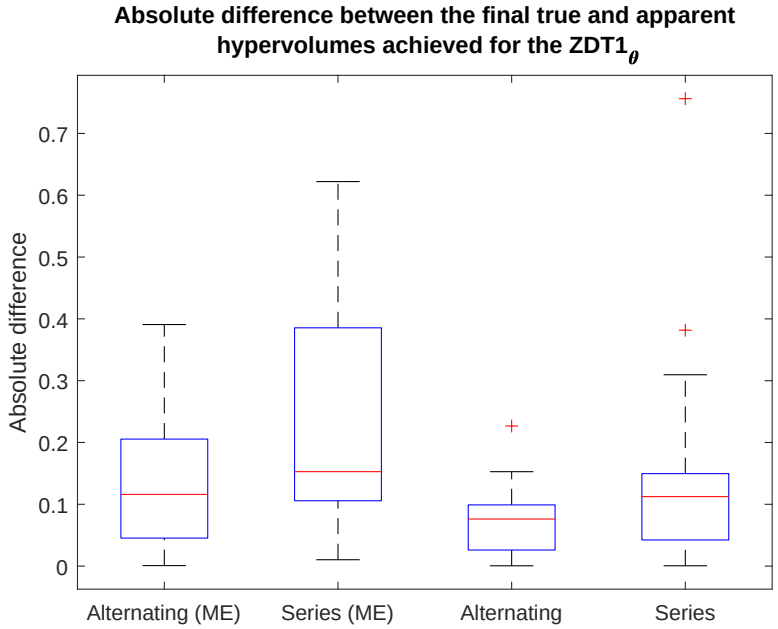
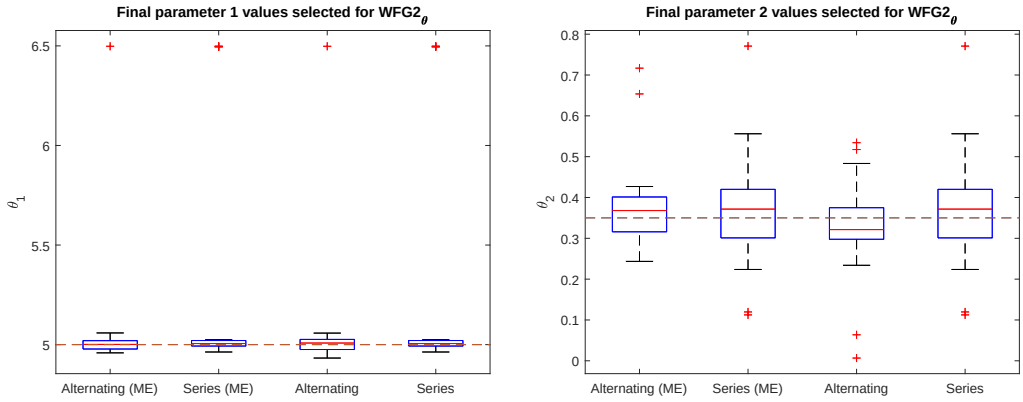


Figure 6.16: Comparison of the absolute difference between the apparent and true final hypervolumes achieved by the alternating and classical (series) methodologies both for the cases when modelling error (ME) is present and absent when considering the ZDT1$_\theta$ function. A lower absolute difference between the hypervolumes was achieved when the alternating method is used.

6.3.3 WFG2$_\theta$ function

Examination of the final parameter values obtained by the two methods when testing with the WFG2$_\theta$

The calibration of the model parameters displayed in Figure 6.17 appear to have achieved good final values especially in the case of parameter 1. It is interesting to note that, for parameter 1, both methods obtained points lying close to its upper limit. What is odd about this, is that there is a distinct separation with a large region void of any points. Looking at the WFG2$_\theta$ function closely, parameter 1 is seen to be located within a cos function. This would explain the points located near the upper limit and implies that the parameter range selected is inappropriate. Now looking at parameter 2 the final values obtained are relatively consistent between the two methodologies, with the alternating method appearing to have a slightly smaller spread of points.



(a) Parameter 1 (b) Parameter 2

Figure 6.17: The final parameter values achieved by the alternating and classical (series) methodologies for both the cases when modelling error (ME) is present and absent when examining the WFG2$_\theta$ function.

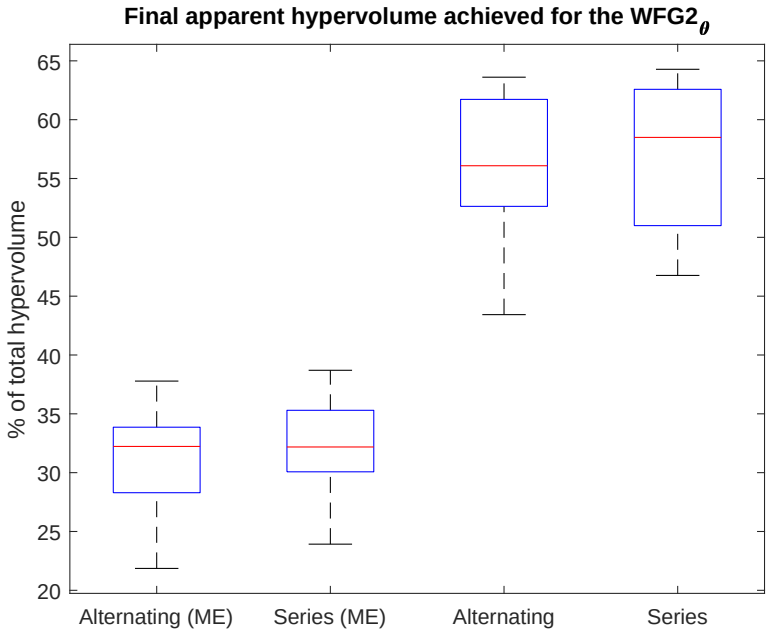


Figure 6.18: Comparison of the final apparent hypervolume achieved by the alternating and classical (series) methodologies, for both the cases when modelling error (ME) is present and absent. The y value shows the percentage of the optimal hypervolume which would be achieved using a reference point takes as the worst case for each objective in combination with the true front.

Comparison of the final apparent hypervolumes achieved by the classical and alternating methods when testing with the WFG2$_\theta$ function

It is evident from Figure 6.18 that the model error has a large impact on the performance of the optimization. From examine the impact on the hypervolume caused by the model error when using the true Pareto fronts, it was determined that this would account for the difference observed within the results. Considering the case were model error is present and absent separately, the performance observed by the two methods is comparable. Similarly to ZDT1$_\theta$ both the alternating and classical methods struggle to achieve the true hypervolume. This can again be accounted for by the larger number of control inputs present within the WFG2$_\theta$ function.

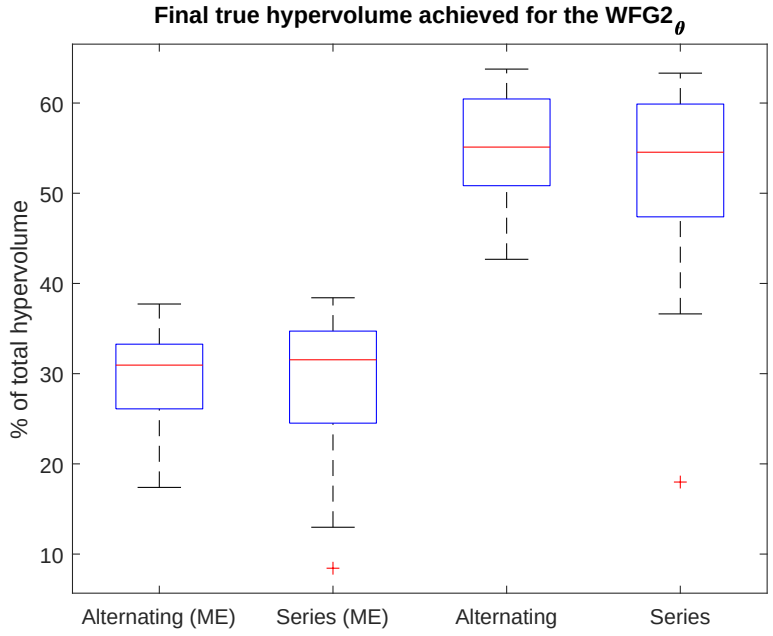


Figure 6.19: Comparison of the final true hypervolume achieved by the alternating and classical (series) methodologies when examining the ZDT1$_\theta$ function, for both the cases when modelling error (ME) is present and absent. The y value shows the percentage of the optimal hypervolume which would be achieved using a reference point takes as the worst case for each objective in combination with the true front.

Comparison of the true hypervolumes achieved by the classical and alternating methods when testing with the WFG2$_\theta$ function

The final true hypervolumes achieved by the alternating and classical methods, displayed in Figure 6.19, shows similar performance to that seen in the apparent

hypervolume. The main changes the true hypervolume shows over the apparent are that the series method, when model error is present, possess a larger spread of points. The other change observed is the median performance of the alternating method, with no model error, matches that of the series method and both have achieve a smaller spread of points.

Comparison of the absolute error between the apparent and true final hypervolume achieved for the WFG2$_\theta$ function using the classical and alternating methods

The absolute difference between the true and apparent hypervolumes obtained by the alternating and classical method, Figure 6.20, once again show consistently lower value being achieved by the alternating method. This behaviour is consistent with what was observed for the DTLZ1$_\theta$ and ZDT1$_\theta$ functions.

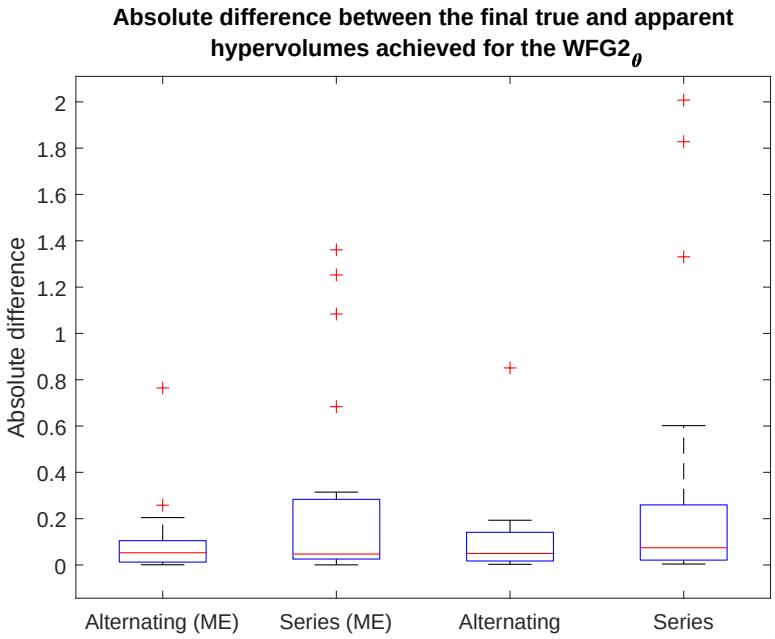


Figure 6.20: Comparison of the absolute difference between the apparent and true final hypervolumes achieved by the alternating and classical (series) methodologies both for the cases when modelling error (ME) is present and absent when considering the WFG2$_\theta$ function. As with the other test problems a lower absolute difference between the hypervolumes was achieved when the alternating method is used.

6.4 Conclusion

This chapter presents a new alternating methodology that employs 5000 evaluations and compares it to the classic approach of performing calibration followed by subsequent optimization.

This novel (alternating) approach employs alternating stages of model calibration and optimization, in order to more efficiently use the available function evaluations. Additionally, the new approach evaluates interim points on the true system between alternations, rather than fully expending them in the initial calibration.

The performance of the novel method is evaluated based on both final parameters obtained and the final hypervolume achieved by each run of the methods, both when model error is present and absent. To ensure a fair comparison, 21 iterations of both the classical and alternating approach are carried out. The statistical significance of the results is assessed using the p-value with the commonly accepted threshold value of 0.025, due to two tails.

Analysing the results for all three test problems, the performance of the novel alternating approach is superior (or at least comparable) to that of the classical approach. There are a selection of hypothesizes for why this could occur, these are: (1) Some thetas are more important than others to get right in the neighbourhood of the Pareto front, so alternating allows a progressively more concentrated calibration; (2) For some problems, it may be easier to get thetas right in a local region (e.g. the Pareto front) than across the search space, so by getting near the PF we make the calibration problem easier. (3) The optimizer is sometimes more badly fooled by some bad calibrations than others. In the case of all three hypothesizes it should be possible to determine their validity through further testing.

The alternating methods shows a consistent improvement in the parameter values obtained during all three of the test problems. The absolute difference between the apparent and true hypervolumes also shows the impact that the alternating method has, with consistently smaller values being obtained. This indicates that while the alternating method may not significantly improve the final hypervolume, it is effective at removing the discrepancy between the true and modelled outputs of the system. It should be noted that both methods are allocated an evaluation budget of the same size, ensuring that any observed improvement is due to the change in methodology. While the methods both use sampling from a distribution, this should not be capable of causing the consistent improvement observed. Additionally there is not currently anything in place that directly accounts for model error, which could hinder their performance.

Having demonstrated that the alternating strategy can prove beneficial when making use of a limited budget, the next step is to reduce the number of available function evaluation from the current 5000 to something closer to a realistic size. In order to do this, it is necessary to develop methods that make more efficient use of the data available. Such methods allow for a greater sharing of information (e.g. through using surrogates) and are examined in the next chapter.

Chapter 7

Architectures for up to 800 evaluations

7.1 Introduction

After looking at problems using 5000 function evaluations the next step is to look at a more limited number. For this work it was decided upon to use 800 evaluations which is a more respectable number when working with expensive problems. In addition, to being more realistic for real world cases with, this decrease in the number of function evaluations being used it is even more imperative that each of them is being used as effectively as possible. In order to achieve this the work, Chapter 7 focuses on the use of surrogate models. Surrogates should allow for better decision making of where to sample new points when using function evaluations.

In Section 7.2 the methodology for using a surrogate model for solving the combined problem is laid out. First an overall look at the main control function is given with a brief explanation of the different sections before it moves on to look at the surrogate model being used. Changes to the optimization and calibration processes are presented before a detailed plan laying out at what stages different forms of information are available and used is given.

Section 7.3 lays out how the testing of the new surrogate combined methodology is carried out. The different choices that were made pertaining to method selection are explained.

Section 7.4 lays out the results from running both the new alternating method as well as the classical method using the methodology which was laid out in the previous sections. Before presenting the results of either method, the determination of methods internal parameters is first examined and the initial data that is used for both methods is presented.

7.2 Combined solution

The newly proposed methodology designed to tackle the combined problems of model calibration and optimization with only a limited budge of around 800 evaluations draws upon concepts from both the ParEGO method, discussed in Section 2.3.8, as well as the Bayesian calibration detailed in Section 6. The main idea that is added with this new method is the creation and use of a shared population of points compiled from the function evaluations performed over the cause of the run. This population aims to prevent knowledge from being wasted and is used for the creation of surrogate model (also known as meta-models).

When designing the new method, the process for running it was broken down into multiple sections. In order to make this work a main controller was estab-

lished to set up the various control parameters for the methodology, as well as obtaining the initial data and calling the necessary steps that need to be taken. The pseudo code for this main process is presented within Algorithm 7.1.

Algorithm 7.1 Pseudo code for main file of combined methadology

```
Set up workspace
Define run order
Select test function form the model and system
Retrieve function information
Set up variable for - General code, Surrogates, calibration, optimization
Set up seed for random number generation
for 1 : Number of replications do
    Initialise control inputs
    Initialise expert inputs
    Initialise parameters
    Retrieve modelled outputs
    Retrieve expert outputs
    Obtain weightings
    Set up CurrentData
    for i = 1 : Number of run steps do
        switch Run order do
            case Optimization
                for 0 : Number of run steps (i) do
                    Run optimization
                    Add in new expert points
            case Calibration
                for 0 : Number of run steps (i) do
                    Run calibration
Save stored data
```

It is possible to break the pseudo code presented in Algorithm 7.1 down into five main sections. In order these sections are:

- **Set up the algorithm run** - During this initial step the different elements of the algorithm are setup with the main objects that will contain variables being defined. A couple of the major choices that should be noted here are the selection of the run order and how many iterations of each stage will be performed. This is also the point at which the test function is selected and the corresponding information is retrieved.
- **Initialization** - The next stage of the algorithm is to initialise the parameters and inputs. As described in the previous chapter these values are obtained using Latin hypercube sampling. Once the initial values are gathered the corresponding outputs are obtained through evaluating the initial input and

parameter set on the function. the expert outputs are also obtained through evaluating the inputs with the defined true parameter values.

- **Store values within objects** - In order to simplify the management of variables within the implementation they are stored within a set of data objects that can be passed more easily between functions.
- **Determine run stage** - Now that the algorithm is set up and the initial data has been obtained the framework determined which of the processes, either optimization or calibration, will be run dependent on its progress through an array called 'Run_order'.
- **Acquire additional expert point** - If optimization was run then a point is selected from the newly obtained undominated points. This newly selected point is evaluated using the true parameters and added to the expert population. Depending on the number of expert points available it may not be possible to obtain a new expert point after each execution of the optimizer. In such cases it would be necessary to add additional constrains on when a new point would be obtained for the expert population.

The next sections fist go through the modelling process, Section 7.2.1, and an examination of the optimization and calibration stages, Section 7.2.2, before going on to look at tracking information throughout the process, Section 7.2.3.

7.2.1 Kriging model

The largest difference between this new method and the one presented in the previous chapter is the inclusion of surrogate models. The purpose of utilising surrogate models is to allow for computationally cheap evaluation which can aid in guiding where the more expensive model evaluations will be carried out. Doing this has the aim of improving the effectiveness of each of the evaluations with the aim of obtaining better performance when working with more limited resources.

After consideration, the kriging model was selected for use within this method due to it having been previously shown to be effective for use when dealing with expensive problems within methodologies such as the ParEGO algorithm. A brief overview of the kriging model (a type of Gaussian process) was given within the literature review, Chapter 2, and went over both what it was as well as a selection of the different variations of the surrogate that are present within the literature. While there are advantages to using kriging there are also potential drawbacks with using a kriging operator for the surrogate model. One of these problems is the internal requirement of the kriging operator to perform a matrix inversion. While there are strategies that can be used to reduce the computational expense

of doing this it remains a factor that should be considered.

The surrogate model is set up to be over both the control input and parameter space. When looking at the kriging model from the perspective of implementing it for use as a surrogate the process can be broken into two parts, the construction of the kriging model and calling the kriging model to evaluate a new set of inputs. The pseudo code for the surrogates construction can be seen in Algorithm 7.2.

Algorithm 7.2 Pseudo code for kriging surrogate setup

1: **Inputs :** *Inputs, Outputs, code, Surrogate_setup*
2: **Outputs :** *Surrogate*
3: **Function :** *Kriging_surrogate_setup*

4: *Set up function(based on the likelihood function that is to be minimised)*
5: *Run a genetic algorithm to determine the best value of 'θ_surrogate' for within the surrogate*
6: *Calculate the Correlation matrix R using the best θ_surrogate and the Inputs*
7: *Find the inverse of R*
8: *Determine the mean*
9: *Determine the standard deviation*
10: *Store all data within the object 'Surrogate.'*
End Function

This function constructs the surrogate using the training data that is passed to it and stores the required data to run the surrogate within the output object 'Surrogate'. The *Inputs* and training data both consist of vectors containing both parameters and control inputs, $[x, \theta]$. There are four main components that are carried out in the surrogate creation.

- **Kriging parameters** - The kriging surrogate model possesses two internal parameters, p and θ, which need to be discovered before it is possible to determine the correlation matrix. The effects of varying the internal parameter p can be seen in Fig. 7.1. As the value of p increases the corresponding correlation becomes smoother while if the value of p is small it implies that there is no immediate correlation. The results of changing the second internal parameter θ can be seen in Fig. 7.2. Low values of θ will result in all points having a high level of correlation while small values of θ can lead to only minimal amounts of correlation between points.

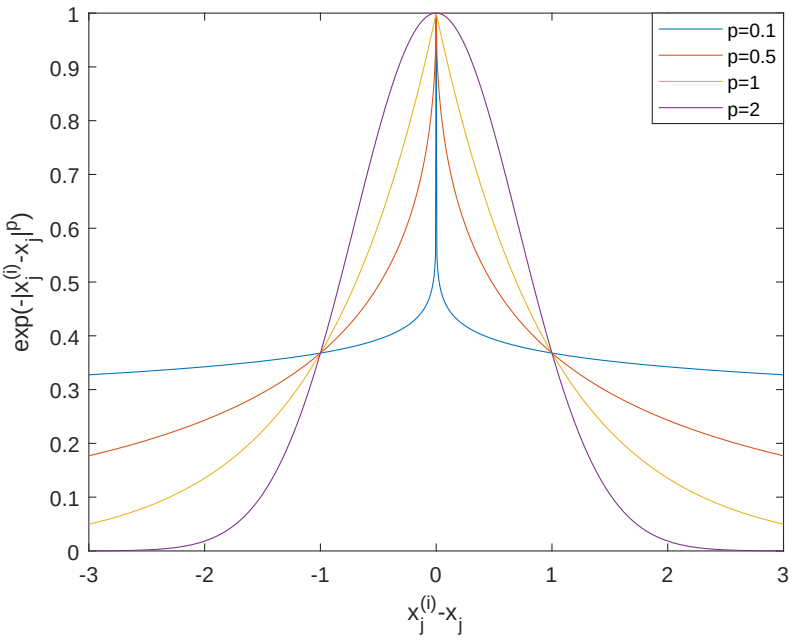


Figure 7.1: Effect of varying p on the correlation, with x axis showing distance from sample point and the y axis showing the correlation. A p value of 2 is often chosen for use within the liturature.

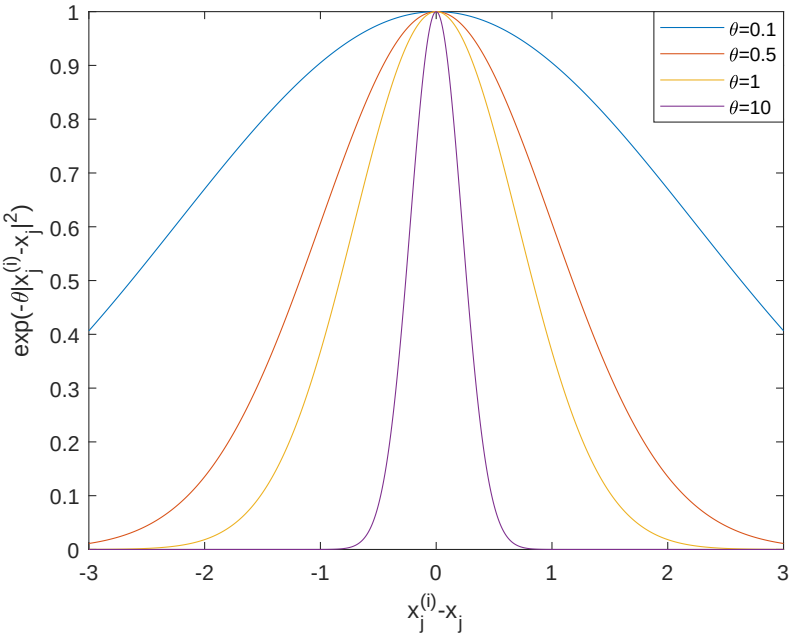


Figure 7.2: Effect of varying θ on the correlation, with x axis showing distance from sample point and the y axis showing the correlation. An appropriate value of θ for each input within the training data will need to be found.

For this work it was assumed that the value of p has been maintained at 2 while the value of θ needs to be determined each time the kriging model is built. Line 4 and 5 in Algorithm 7.2 refer to the determination of the value of θ. Each of the inputs to the surrogate model can possess a different value of θ which is dependent on the data being used to build the surrogate. In order to determine the values of θ a function that finds the likelihood for a given set of data and a selected θ is first constructed. This function is then used within a genetic algorithm to determine appropriate values of θ.

- **Correlation matrix** – Now that the internal parameter θ has been determined, and p has been taken as 2, it is possible to calculate the correlation using Equation 7.1

$$cor[y(x^{(i)}), y(x^{(l)})] = exp\left(-\sum_{j=1}^{k} \theta_j |x_j^{(i)} x_j^{(l)}|^2\right) \tag{7.1}$$

Through calculating this for each of the elements it is possible to construct the correlation matrix. In order to improve the computation time of the code instead of calculating each element of the correlation matrix only the upper half of the correlation matrix was constructed. Using the upper half the full correlation matrix could be constructed using Equation 7.2

$$Correlation = upper + upper' + eye(n) + (eye(n)eps) \tag{7.2}$$

Where $upper$ is the upper half of the correlation matrix, eye is an identity matrix, n is the number of input points and eps is a small number. The purpose of adding the, $(eye(n)eps)$ is to help reduce the chance of the created correlation matrix being ill conditioned.

- **Cholesky factorization** – The next step after the determination of the correlation would be to acquire its inverse. While, when working with small population of points this may not pose an issue, it can prove to be prohibitive with larger populations due to its high computational cost. In order to avoid the need to calculate the inverse matrix, Cholesky factorization was used with back substitution. For its implementation the **chol()** function from MATLAB was used.
- **Mean and variance** - The final stage of constructing the surrogate model, shown in lines 8 and 9 of Algorithm 7.2, is to use back substitution to calculate the mean and variance.

$$\mu = \frac{one' * (U \backslash (U' \backslash y))}{one' * (U \backslash (U' \backslash one))} \tag{7.3}$$

The mean is calculated as shown in Equation 7.3, in which y represented the training data outputs, U is the Cholesky factorization of the correlation matrix and one is a matrix the same dimensions as y in which all values had been set to 1. Once the mean had been found it was substituted into Equation 7.4 to find the variance. The only additional variable n represents the number of training points present.

$$\sigma^2 = \frac{y - one * \mu' * (U\backslash(U'\backslash(y - one * \mu)))}{n} \tag{7.4}$$

The function displayed in Algorithm 7.3 takes in a set of new points along with the object surrogate that contains all the relevant information to a constructed kriging model and returns the surrogate output along with the mean squared error (MSE) of the prediction.

Algorithm 7.3 Pseudo code for kriging surrogate

1: **Inputs :** *New_input,Surrogate*
2: **Outputs :** *Surrogate_output, MSE*
3: **Function :** *Kriging_surrogate*

4: **for** $i = 1 : length(New\_inputs)$ **do**
5: *Calculate the correlation between the new input points and surrogate input points*
6: *Calculate the surrogate output*
7: *Calculate the mean squared error of the prediction*
End Function

The correlation between the new point being looked at and those present within the surrogate (represented as r) is calculated using the formula from Equation 7.1. Once the array containing the correlation for the new point has been created it is used to calculate the output, $\hat{y}$, for the given input.

$$\hat{y} = \mu + (r' * (U\backslash(U'\backslash(y' - (one * \mu))))); \tag{7.5}$$

$$MSE(i) = \sigma^2 \left(1 - (r'(U\backslash(U'\backslash r))) + \frac{((1 - (one')(U\backslash(U'\backslash r))))^2}{one'(U\backslash(U'\backslash one))}\right); \tag{7.6}$$

It is also possible to calculate the mean squared error of the predictor using the information that has been obtained up to this point. The formula for doing so is presented in Equation 7.6. For both of these equations the Cholesky factorization found during the building is used with back-substitution in order to avoid the need to perform matrix inversions. If matrix inversion were to be used then the

inverse correlation matrix, R^{-1}, could be substituted in as seen in Equation 7.7.

$$R^{-1}y = (U \backslash (U' \backslash y)) \tag{7.7}$$

7.2.2 Updated optimization and calibration

Optimization

As mentioned earlier the optimization method used for this new method is based on the ParEGO algorithm. The choice was made due to the ParEGO algorithm inherently already incorporating the use of a surrogate models to deal with expensive problems, as mentioned within the literature review in Chapter 2.3.8.

Similarly, to the MOEA/D algorithm, ParEGO is also a decomposition-based method that looks at breaking down more complex high dimensional problems into a set of simpler subproblems. While both MOEA/D and ParEGO use a set of weight vectors to define the separate subproblems, ParEGO selects the one currently being worked on uniformly at random, which is unlike MOEA/D that cycles through the different subproblems. After implementing the ParEGO algorithim in its original for it was found that there were certain aspects of it which needed modification to work within the combined method. The pseudo code for this altered optimization method can be seen in Algorithm 7.4.

The implementation of the optimization, as shown in Algorithm 7.4, begins by following the ParEGO algorithm with the reference direction being selected uniformly at random from the possible reference directions. The selected reference direction is then used to get the scalarised outputs for points within the population. The first main change comes before constructing the surrogate model, where it is necessary to determine which points within the population will be used as training data. Ideally all the points would be used, however this can be very time consuming, so the population is limited. The two main cause of the increased run time is the need to compute the correlation matrix as well as perform a matrix inversion when constructing the surrogate. Both processes are highly computationally expensive and while it was possible to avoid computing the inverse directly the correlation matrix still needed to be found. The reason for this high computational expense is that with each new point added it is necessary to compare it with all the other points which have been included.

The method by which the limited population, that is used to build the surrogate, was selected within the ParEGO algorithm was to select half the points based on those with the smallest scalarized output followed by selecting the remainder at random from the remaining population. There are many other criteria upon

Algorithm 7.4 Pseudo code for optimization

```
Function : Optimization

Select reference direction uniformly at random
Get scalarized outputs of data using current reference direction
Determine which points will be used to construct the surrogate
Obtain the surrogate model for each objective using 'Kriging_surrogate_setup'
Get scalarized outputs from current population
Acquire temporary population consisting of mutated points and points from a Latin
hypercube
Evaluate points within the temporary population using 'Kriging_surrogate'
Determine the scalarized output for each of these points
Set up parent pairs for the evolutionary algorithm
while Current EA iteration < total EA iterations do
    for 1 : Number of parent pairs do
        Perform crossover
        Perform mutation
        Ensure child lies within limits
        Find the surrogate output of the child using 'Kriging_surrogate'
        Find the scalarized output of the child
        If the scalarized output of the child is superior to that of the parent replace it
Combine the parent population
Select the best point
Add new data to Current_data
End Function
```

which points could be chosen and a list of some of the considered criteria can be seen below,

- Highest actual improvement,
- Highest EI
- Newest points
- Smallest scalarized output
- Random

Due to the points being used needing to model both the control inputs and parameters instead of just the inputs such as with ParEGO it was proposed that other point selection schemes could prove beneficial. The final selection method decided upon for building the surrogate for optimization took half the available points from those with the smallest scalarized output for the given reference direction. Instead of selecting all the remaining points at random, half of the remaining points were chosen based on those which show the highest actual improvement. This measure is determined by calculating the difference between the scalarized

output of the surrogate and the model for a given point. These points should identify regions in which the surrogate has the largest amount of modelling error and aid in combating it. The remaining points comprising the build points for the surrogate are selected at random from the points which had not already been selected.

It should be noted that before the selection of point for construction of the surrogate was carried out the possible population was refined. This was done in order to remove any duplicates or points which were close to identical to prevent computation errors. Points that were close to identical were defined as those who would possess the same inputs if they were rounded to three decimal places.

Once the training population is selected, the surrogate model is constructed, this is done using the function detailed in Section 7.2.1. Next, as was done in ParEGO a parent population consisting of 20 points is constructed. Five of the points were chosen from current population depending on which possess the smallest scalarized outputs under the prevailing reference direction. The remaining 15 points are selected through Latin hypercube sampling of the input space.

The Second major difference between this newly constructed method and ParEGO is that the expected improvement (EI) is not used as a criterion for selecting new points, instead the scalarized output is used. This change was made as it was found that when EI was being used there were many cases where most points ended up focusing on selecting points which help to remove the error. While improving the model is not a bad thing in those cases there was very little progress made towards discovering the actual front making the use of EI unviable.

Returning to the algorithm, once the parent population is obtained the corresponding surrogate outputs are evaluated and scalarized outputs determined. The parent population is ordered based on their scalarized output and broken down into parent pairs for use within an internal genetic algorithm. The code then loops through a pre-set number of iterations. During each of the loops singular binary crossover is performed on the parent pairs before a mutation is applied. Once the new child point is created it is assessed to ensure that its inputs lie within previously stated boundaries. If the inputs are determined to lay outside of these boundaries, they are shifted to the boundary value.

The child point is then evaluated using the surrogate model and its scalarized outputs, dependent on the current reference direction are obtained. The scalarized output of the child is finally compared against that of its superior parent. If it is found to have performed better, it replaces its parent and if not, it is discarded. This process is repeated till the whole evaluation budget has been used up at which point the parent population is compiled and the best point from the

optimization is determined. The new data is added to the corresponding data objects.

Calibration

While the methodology for calibration has remained mostly the same as that seen in Section 6.2, a major change has been made where, before when the function would have been called the surrogate is now called instead.

Similarly, to the optimizer which was just discussed a population of points needs to be determined for training the surrogate model. The requirements of the model are different however as during optimization the surrogate focused on the reference direction while for calibration the region in which the best parameters may lie is not known. Due to this the types of points selected have been adjusted slightly.

The first set of points obtained for the training set comprising of a quarter of the surrogate build population are again taken based of the points possessing the best scalarized output although this was calculated using a weighted sum with equal weighting rather than the Tchebycheff function. These points aim at giving us information about the region near to the Pareto front. The next quarter points are selected based on those with the highest actual improvement followed by a quarter of the points being selected based on the lowest actual improvement. These are chosen to give information about regions with high levels of error as well as maintain information about good regions. The final quarter of the points are selected by random sampling out of the remaining points. Before commencing this sampling the population is again processed to ensure that identical or close to identical points have been removed.

The 'Kriging_surrogate_setup' function is called to determine the components of the surrogate as it was within the optimizer. Once the initial MCMC point has been chosen it is evaluated on the newly constructed surrogate model and its likelihood is calculated with the resulting value being stored. The MCMC is then carried out as previously described in Section 6.2.

After the MCMC has finished running a new set of parameters needs to be extracted from the run. There are many methods by which this could be done such as fitting a kernel to the results after burn in or selecting some type of average. For this work however the selected parameters are simply chosen based on the ones which managed to achieve the highest likelihood.

A more thorough breakdown of the calibration process used can be seen in the pseudo code for Algorithm 7.5.

Algorithm 7.5 Pseudo code for model calibration

```
1: Function : Calibration

2: Select points to be used within surrogate
3: Obtain the surrogate model using 'Kriging_surrogate_setup'
4: Initialize MCMC chain
5: Assess the current model parameters to find the corresponding model outputs using
   'Kriging_surrogate'
6: Get scaled expert output
7: Calculate likelihood
8: Determine/ set the Prior distribution
9: for 1 : loop per parameter do
10:     for 1 : number of parameters do
11:         while Candidate point is out of bounds do
12:             Produce candidate point
13:             Check to see if candidate point is within bounds
14:         Determine model outputs using 'Kriging_surrogate'
15:         Get the scaled expert output
16:         Determine the likelihood of scaled model outputs
17:         Calculate posterior from likelihood and prior
18:         Determine if the new point will be accepted or not
19:         Store relevant data
20: Determine the newly accepted set of parameters
21: Calculate the model output and Likelihood of the best current parameters
22: Calculate the scalarized output of the best points
23: Store relevant data
    End Function
```

7.2.3 Tracking information throughout the combined method

Now that an overview of how the algorithm operates has been given it is important to have a clear understanding of when different types of data are obtained and used. There are three main types of information being acquired throughout the time the algorithm is running. These are expert/real world data point, modelled data points and data points from the surrogate. Each of these types of data will only have a limited number of points used throughout the entire run.

Real world data points are highly limited with only around ten points being used. The modelled points are also limited with around 800 points being used. Finally, while the number of surrogate evaluations is not specifically limited the total amount used will depend on the number of times that the optimizer and calibration are called as well as the run limits are for their internal GA and MCMC. The information spread shown below is looking at the case of the alternating method in which the algorithm goes back and forth between the calibration and

optimization.

Initialization

- Part of the real-world evaluation budget is used (E.g. around 5 points)
- Around 100 model evaluations are used to create the initial output population. This population consists of points with differing control inputs and parameters.

Calibration

- No additional real-world evaluations are used. The expert points already gained are used within the likelihood calculation.
- After the MCMC has been fully run the parameter set that was determined to be the best is evaluated on the model in conjunction to the expert inputs. This means that the total number of function evaluations used for each calibration should be in a range of about 5 to 10 evaluations.

Optimization

- After all optimization runs in a batch have been performed a single point is selected from within the undominated points to be evaluated using the expert model. Depending on how many times optimization is called this may need to be limited.
- Each iteration of the optimizer uses only a single function evaluation. The point chosen for evaluation possesses the highest kriging prediction out of the final parent population the genetic algorithm discovers.

If the classical method, in which the calibration and optimization run in series, were to be considered then the expert points would all have been evaluated at the start. This means that while there would have been more model points used in each calibration run there would be no additional expert points being acquired after optimization.

7.3 Algorithm setup

The two methods have been set up to allow for as fair a comparison as possible with both methods having the same amount of computational budget assigned to them. The seed for the random number generation has been fixed for both cases to the same value. This ensures that the initial populations for both runs are the same to ensure that they have a fair starting point and to help remove any possible sources of bias.

7.3.1 Run order

When setting up the algorithm it is necessary to determine the number of times you wish to alternate between performing model calibration and optimization as well as how many iterations of each of them you wish to perform. Currently the method is set up to perform ten iterations of calibration followed by seventy iterations of optimization before going back to calibration. When calculating the total number of evaluations that this will use it is important to note that the number of evaluations used by the model calibration will increase as the size of the expert population increases. In this case if we were to alternate 5 times we would use,

$$Optimization\ model\ evaluations\ =\ 70 * 5 = 350, \tag{7.8}$$

$$Calibration\ model\ evaluations\ =\ 50 + 60 + 70 + 80 + 90 = 350, \tag{7.9}$$

$$Total\ model\ evaluations\ =\ 100 + 350 + 350 = 800. \tag{7.10}$$

The additional hundred evaluations come from those used during the initialization stage. If the classical approach of performing calibration followed by optimization is considered then thirty five calibration loops would be used, as 10 expert set are present. These calibration iterations would be followed by three hundred and fifty optimization iterations to use up the same total 800 evaluations, including the hundred initial evaluations.

As the number of expert points is limited it is assumed that there are only ten expert evaluations available over the cause of the entire algorithm run. The available expert evaluations are split into two sets with the first five being used during initialization and the remaining five used after each of the batches of optimization runs are completed.

7.3.2 Surrogate parametes

The internal parameters of the kriging model need to be determined in order to create the surrogate. When trying to determine the value of the internal parameters (θ), we assume that they should lie within a reasonable range of possible values. For our work the range is set as 10^2 to 10^{-3} which is converted to searching between 2 and -3 for the genetic algorithm during surrogate creation. If the value of theta lies below 10^{-3} this indicates that it has very little impact on the output of the surrogate. The results of an experiment to see how many runs of the GA need to take place for a suitable value of θ to be obtained can be seen in

Section 7.4.2.

7.3.3 Calibration and optimization parameters

Both the Calibration and optimization contain internal parameters that need to be set before it is possible to run the combined method. These values can be seen in Table 7.1 and 7.2.

Parameter	Setting
Surrogate size limit	150
Number of scalarization vectors	11
Scalarizing function	Tchebycheff
Internal GA evals per iteration	1000
Tournament size	20

Table 7.1: Internal parameters of the optimizer

Parameter	Setting
Surrogate size limit	150
Number of scalarization vectors	1
Scalarizing function	Weighted sum
MCMC Burn in	30%
MCMC step	1000

Table 7.2: Internal parameters of the calibration

7.4 Results

Using the setups detailed in the previous section, the algorithm was run for both the series and alternating cases. The results are detailed in five sections; firstly, the setup details are discussed, flowed by looking at the initial data, then the results of using the alternating method are discussed, subsequently those generated from the classical series method are examined and finally results directly comparing the two methods are looked at.

7.4.1 Selecting the appropriate number of generations

Before performing either setup, the minimum number of generations required by the internal Genetic algorithm (GA), utilised in the generation of the surrogate model, to produce an adequate fitness value is determined. While it is possible to skip this step, and simply select a large number of generations for the genetic

algorithm, it can lead to a number of problems. For example, too few generations can result in the final θ value selected by the genetic algorithm being suboptimal and causing the surrogate model to underperform. Conversely, an excessive number of generations can lead to an increase in the overall run time of the algorithm, causing it to be increasingly computationally expensive, especially as surrogate creation is perhaps the most computationally expensive step within the algorithm. The results of running the genetic algorithm for fifty generations are shown in Fig 7.3.

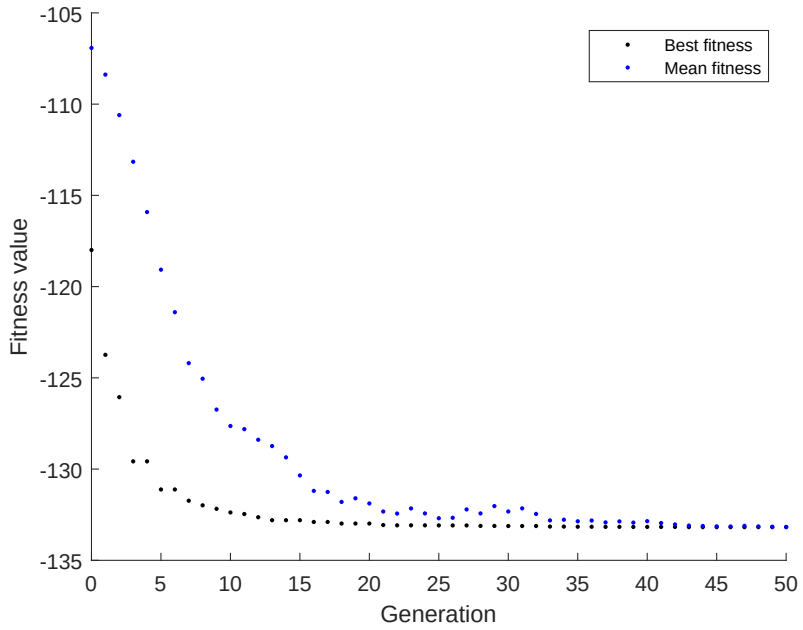


Figure 7.3: Fitness change within a GA searching for best theta during surrogate construction. Twenty five generations were selected for use within the GA as this was the point at which both the best and mean fitness are close to the minimum fitness found.

From Fig. 7.3 it can be observed that after approximately 20 generations the rate of improvement of the 'best fitness' falls off and the marginal rate of improvement approaches zero. There is an inherent variance between runs performed in a generation, so generation's mean run and best run will therefore differ. The relative distance between the mean and best performance can be used as a gauge to identify convergence towards the best run and aid in identifying the diminishing marginal returns from continuing the search for an appropriate theta. The number of generations selected for the GA within the surrogate build is 25 as this is when the marginal return from increasing the number of generations becomes insubstantial.

7.4.2 Initial evaluations

Before discussing the progress and performance of the alternating method, the initial information available to the algorithm is detailed. Fig. 7.4 depicts the different Control inputs (Ci), Parameters (P) and the Model outputs (MO) available to the algorithm at the start of the run.

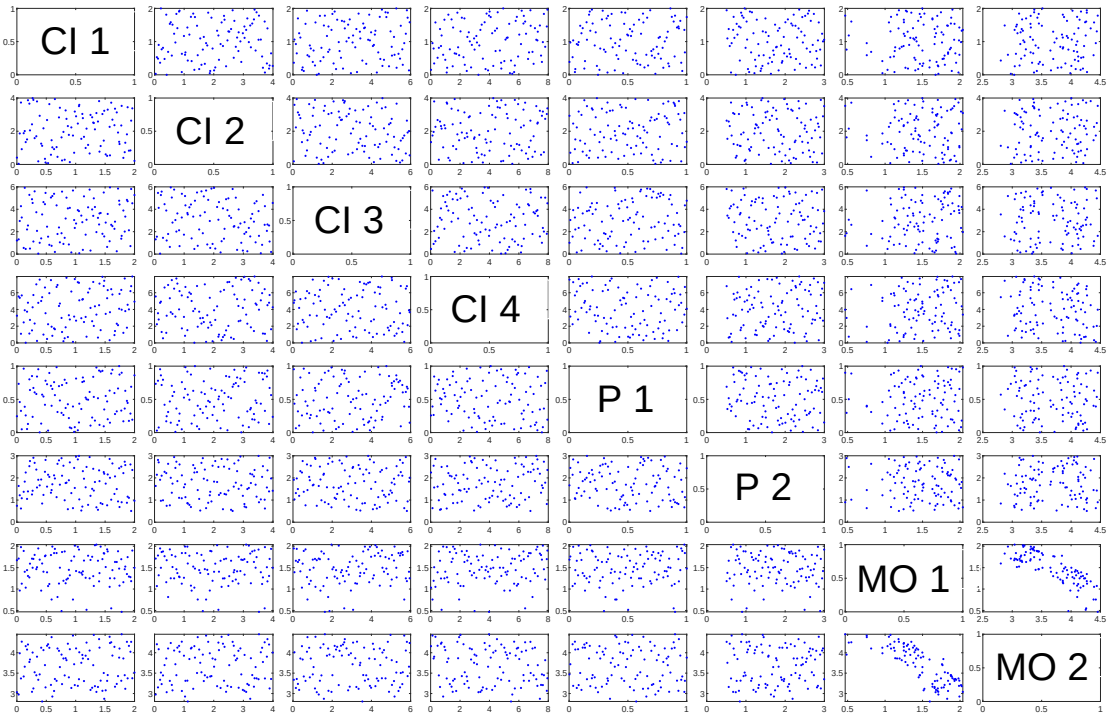


Figure 7.4: The control inputs, parameters and model outputs corresponding to the points present within the initial population. A good degree of coverage has been achieved for all the control inputs and parameters, with only small regions having a lower density of initial points.

Fig. 7.4 shows that by selecting varied control inputs and parameters, via the use of Latin hypercube sampling, a wide spread of points is obtained. For the most part, there is excellent coverage of the majority of the input domain for each control input and parameter. There are a few regions for which using an initial population of 100 points still shows some areas that are sparsely populated (such as in the central area of Ci3 , P1). The majority of the model outputs (MO in Fig. 7.4) exhibit no significant correlation between any of the control inputs or parameters. The only clustering that can be see is present within the model outputs for which higher values appear to be obtained more often.

7.4.3 Results from the alternating method

The first aspect of the alternating method discussed is the results produced by the model calibration over fifty iterations. During each iteration MCMC is performed and the parameter set possessing the highest likelihood is selected to be evaluated on the model and added to the point population (see Section 7.2.2). These selected points are displayed in Fig. 7.5.

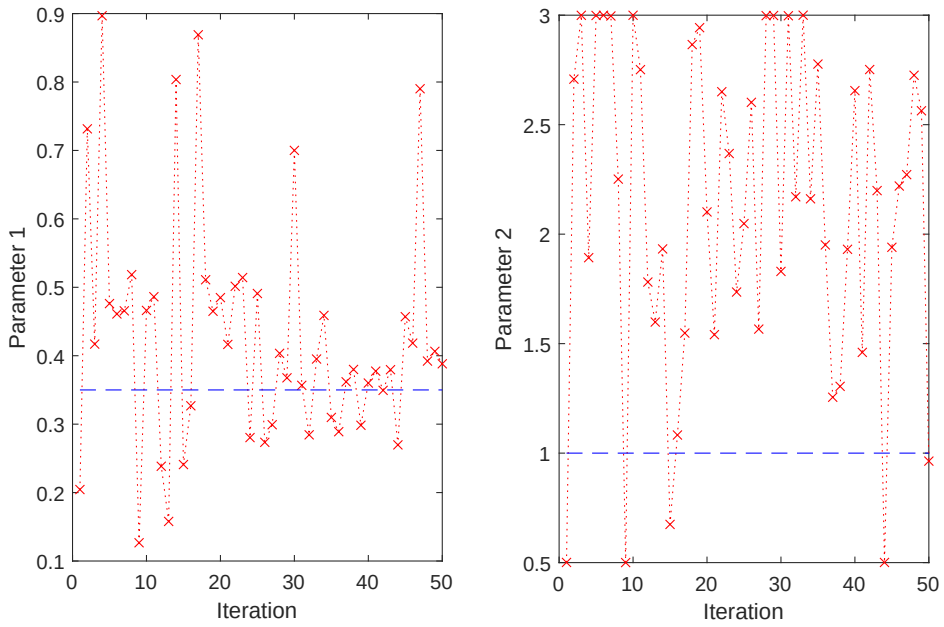


Figure 7.5: The selected parameter from each of the calibration iterations (red cross) with the true parameter value (blue dashed line) shown. As the calibration progresses the selected parameter 1 values move closer to the true value. There is no clear trend in the values of parameter 2 selected.

Fig 7.5 shows the change in selected parameter values over calibration iterations. In this alternating method after every tenth iteration step (10, 20, 30, 40, 50) a new expert point is selected and added to the population being used. The result is that by the end of the iteration sequence the full ten expert points are included in the population.

Evaluating the overall progress of the calibration using the alternating method, a significant improvement in the selection of parameters over the initial values is observed. Firstly, parameter 1 initially has high variability in the selected values, with many of the points being located far from the true value. Such points represent cases in which a local minimum are discovered. This test problem has local minima that the MCMC could potentially get stuck in, located around 0, 0.77 and

1. To get a clearer image of where these local minima occur refer to the problem definition in Section 5.5.2. These three local minima appear to be able to explain many of the points which diverge from the desired parameter values.

Evaluating the calibrations chosen values of parameter 2 it can be seen that there is much more variation in the values being selected than in the case of parameter 1. In general, when a value of parameter 2 is selected which lies close to the true value it corresponds with a value of parameter 1 which also lay close to the true value. It is known that while large changes in parameter 2 are impactful, smaller variations in it have a limited impact on the output. In addition, the impact of varying parameter 2 is also much greater when the values are small.

	Selected parameters				
	1	2	3	4	5
Parameter 1	0.417	0.327	0.327	0.362	0.362
Parameter 2	2.999	1.083	1.083	1.255	1.255

Table 7.3: Parameters selected for use within optimization after each batch of calibration was completed. Selected parameter values progress towards true values. Within some batches there is a lack of improved points being obtained. These results are from a single run of the alternating methodology.

The parameter values selected for use within the optimization after each batch of ten calibration iterations are detailed within Table 7.3. The selected parameters remain the same both between the second and third batches and between the fourth and fifth. This is because no new point possessing a higher likelihood was obtained. The three points that were selected are, in order, the 3^{rd}, 16^{th} and 37^{th} points.

In order to get a clearer idea of how the selected parameters actually effect the model outputs Fig. 7.6 shows how the absolute difference between true and modelled scalarized outputs of the system vary with the selected parameters.

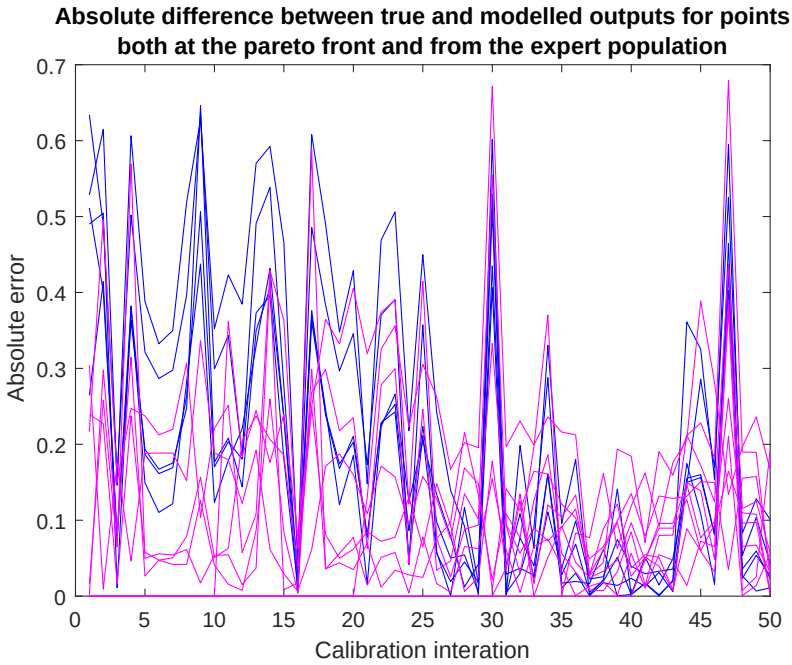


Figure 7.6: Absolute difference between the modelled outputs using the true and determined parameter values. The blue lines show points on the Pareto front and the magenta lines show expert points. A clear trend of reducing error can be seen. In the later iterations the performance of the points located at the Pareto front outperform that of the expert points.

There are two types of points considered in Fig. 7.6, the first of which consists of the points within the expert population at each step of the calibration iteration (shown in magenta). The second type of points considered consist of those which lie along the Pareto front (shown in blue). Fig. 7.6 shows a general decrease in the absolute error over the course of the calibration.

More interesting than the reduction in absolute error, is relative change between the absolute error of the expert and Pareto optimal points. Initially it can be seen that the points within the expert population have a lower absolute error than that of points that lie on the Pareto front. As the calibration progresses however the points lying on the Pareto front show greater improvement until roughly iteration 27 whereby their performance starts to consistently outperform those from points from within the expert population. The large peaks that can be observed throughout the calibration appear to relate to cases where the parameter values lie close to local minima's. The number of such cases also occurs less often later into the calibration by which time the performance is relatively stable.

The first output of the model when a set of predetermined inputs, $x = [0.5\ 0.5\ 2.1\ 2.8]$, were used to assess the impact of varying parameters can be

seen in Fig. 7.7. A pre-determined set of inputs were used to allow for comparison between the different methods. There is a clear impact when parameter 1 is changed. Changing parameter 2 causes variation in the oscillatory of the model output. When small values of parameter 2 are selected there are more peaks and troughs, possessing extreme output values.

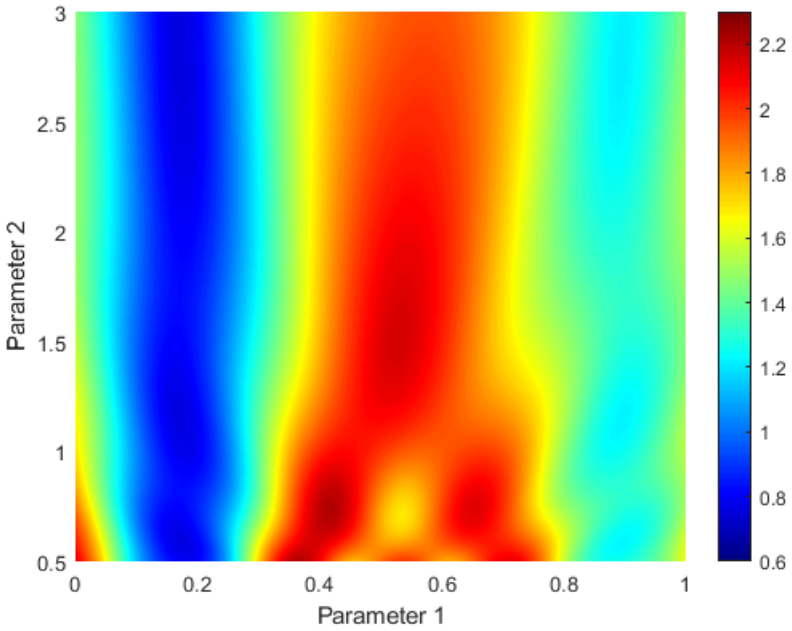
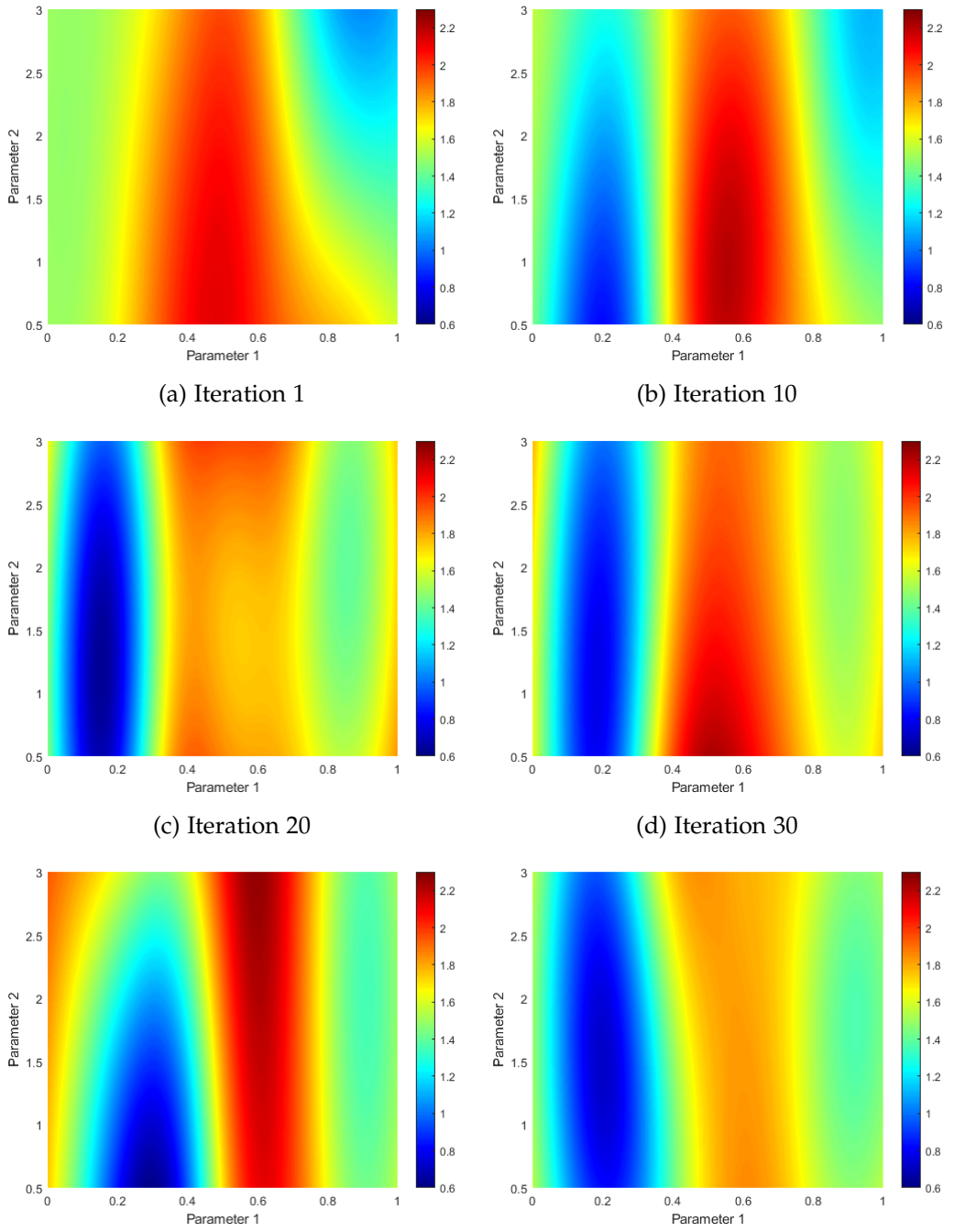


Figure 7.7: Model output 1 for fixed inputs with varying parameters. Most variation is dependent on parameter 1. There is a higher density of peaks and troughs at lower parameter 2 values.

A set of six figures showing the surrogate after different numbers of calibrations runs is depicted in Figure 7.8. The iterations of the surrogate output surface chosen to be displayed begins with the starting surrogate formed from the initial data followed by the surrogate produced at each tenth iteration. These were selected as they represent the final surrogates used for calibration before switching over to perform optimization.

The surrogate changes throughout the model calibration and is depicted at six points in Fig. 7.8. The initial, final and each tenth iteration is depicted. The initial surrogate (depicted in Fig. 7.8.a) is only capable of replicating the general shape of the model with both dips in approximately the right place. The surrogate model is, at this initial stage, unable to model areas that do not follow the general trend, such as the peaks and troughs that appear for smaller values of parameter 2. The surrogate generated after the first set of ten calibration iterations (see Fig.



(a) Iteration 1

(b) Iteration 10

(c) Iteration 20

(d) Iteration 30

(e) Iteration 40

(f) Iteration 50

Figure 7.8: The effects of changing the parameters, after 1, 10, 20, 30, 40 and 50 calibration iterations, on output 1 of the surrogate is shown. Fixed control inputs of [0.5 0.5 2.1 2.8]. Initially a large difference can be seen between the surogate output plot and that shown for the model in Fig. 7.7. At the 30^{th} iteration, subplot d, the shape is a much closer match but worsens by the 40^{th} iteration, subplot e. The areas of higher complexity are never well modelled.

7.8.b) shows significantly improved performance over the initial surrogate and more closely matches the model. This is most evident at low values of parameter 1 where the surrogate now forms a trough matching the model output shown in Fig. 7.7. However, the matching is poorer at larger values of parameter 2, where it still has relatively large output values. By the 20^{th} iteration, see Fig. 7.8.c, the trench present at low values of parameter 1 is much more defined and is now present for all parameter 2 values matching the model. The transitions between the peaks and troughs for smaller values of parameter 2 also now lie in the correct location. However, the size of the output for larger values of parameter 1 is much lower than it should be.

Moving on to look at the surrogate output obtained in the 30^{th} iteration, see Fig. 7.8.d, further improvements can be observed. Both the shape of the surface and the size of the outputs more closely match those present within the model. The quality of the surrogate for the 40^{th} run, observed in Fig 7.8.e, has actually detreated quite considerably compared to those of previous surrogates. The final surrogate produced (see Fig. 7.8.f) for calibration exhibits an significant improvement over the 40^{th} iteration in both definition and locations of the troughs and peaks, although it still does not fully match the model. It is concluded that the surrogate appears to struggle at emulating the more extreme regions of the model even after fifty iterations.

In order to be able to gain a better understanding of how well these surrogates are truly doing the plots of the absolute error between the surrogate and the model output for a fixed input set and varying parameters is produced and can be seen in Fig. 7.9. Each of these plots corresponds to the surrogates displayed in Fig.7.8. When considering these graphs, it is important to take not of the scale as just casually comparing them can be misleading. Over the cause of the first four plots, Fig. 7.9.a to Fig. 7.9.d, a reduction in the error globally can be observed. The plots displaying the surrogate error for 40 and 50 iterations (Fig. 7.9.e and 7.9.f respectively) have a large increase of error which is especially noticeable at the parameter values [0.35 0.5]. The principal regions of error in the surrogate arise at the location of the model's peaks and troughs. This again indicates that the employed surrogate model is unable to properly model the extreme regions of the function.

The underlying cause of poor surrogate performance at extreme spatial regions is twofold. Firstly, none of the initial population of points being used to construct the surrogate lie within these regions. Secondly, the location of these regions varies depending on the inputs being considered and hence can cause them to be more difficult to locate. Considering these points, one avenue in future research

is to investigate possible methods by which points lying in such regions could be efficiently identified and included in the surrogate. Alternatively, the surrogate used could be improved by either adapting the current kriging model to correctly identify such points or examining other potentially viable surrogates for use in the approach.

Now that the results of performing calibration have been presented and the performance of the surrogate examined, we shall now look at the findings from performing optimization. We begin this by presenting the modelled output points which have been evaluated using the parameter values displayed in Table 7.3, which correspond to when they were evaluated.

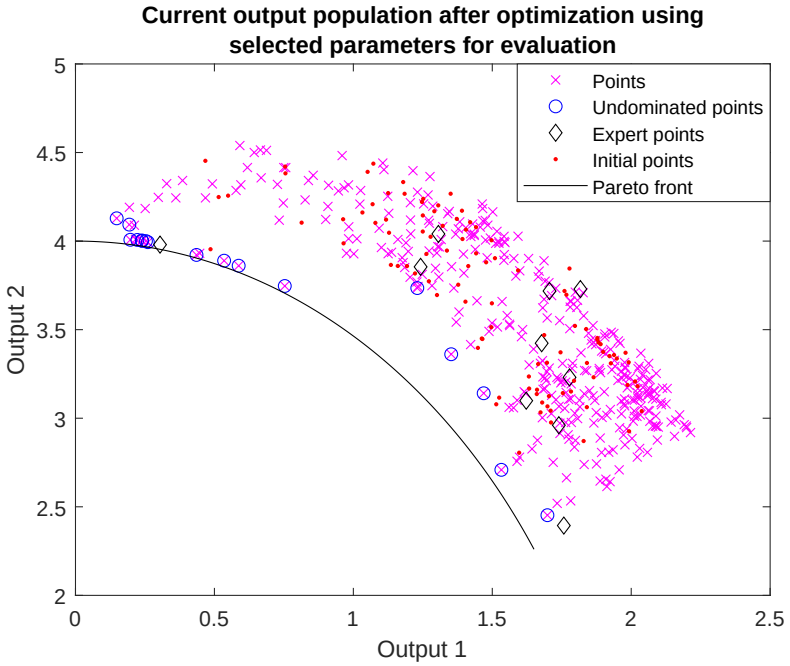
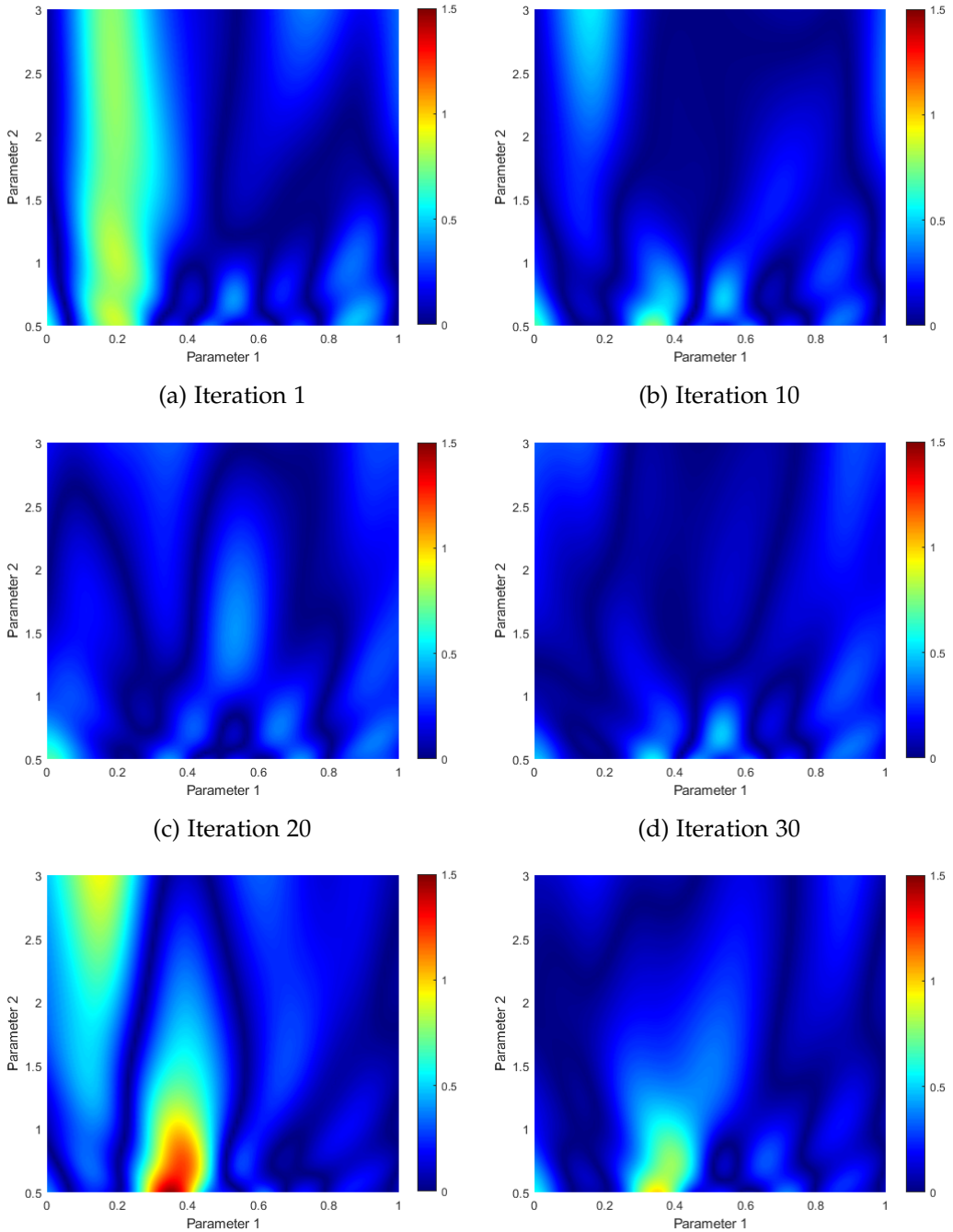


Figure 7.10: The final output space after all optimization runs have been completed evaluated using the selected parameters. Modelled output points are shown as crosses, while the undominated set of points comprising the Pareto front from the population are shown as circles. The set of initial population points are displayed as dots and the expert population are shown as black diamonds. The optimization has successfully found points near the front. It has struggled to identify points at the central region of the output space.

The modelled output from the points is determined using the possible points produced during calibration (displayed in Fig. 7.10). As can be observed, progress is made towards discovering the Pareto front by the optimization especially towards the upper portion of the Pareto front (see Fig. 7.10). During the optimization 11 different reference directions were used and it is possible the see the



(a) Iteration 1

(b) Iteration 10

(c) Iteration 20

(d) Iteration 30

(e) Iteration 40

(f) Iteration 50

Figure 7.9: The absolute error between the surrogate and the model for output 1 after 1, 10, 20, 30, 40 and 50 calibration iterations. Over the first 4 plots the level of error is observed decreasing. There are areas at which peaks and troughs occur in the model at which the error remains high for all stages of the calibration.

optimizers progress along these lines. The clearest example of this the points leading up to the undominated point lying at [1.53 2.71]. The optimization appears to struggle to determine points lying closer to the front in the central region. One reason is the lack of any expert points lying close to the central region of the Pareto front, unlike the lower and upper regions of the Pareto front. With regards to the initial points, only a few approach the front and all of these are successfully dominated by the newly obtained points from optimization.

The same population of points is then evaluated using the true parameters (depicted in Fig. 7.11) . The set of undominated points highlighted corresponds to those produced by the modelled output using selected parameters (Fig. 7.10) is again shown in order to provide a comparison to newly evaluated points.

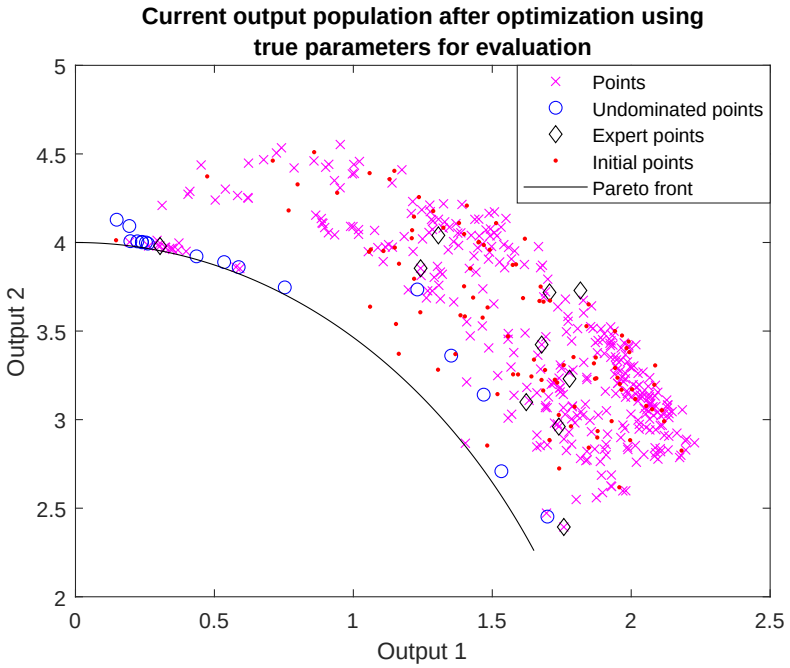


Figure 7.11: The final output space after all optimization runs have been completed evaluated using the true parameters. Points near the front generally either maintain or improve upon their performance when evaluated using the true parameters. Some initial points perform better than believed and move closer to the front.

The true output values produced by the population are comparative to those produced by the modelled output, with some areas appearing to perform slightly better and some performing slightly worse (see Fig. 7.10 and Fig. 7.11). One of the most noticeable differences is the change of the positioning of the initial data points. Quite a few of these initial points which performed badly using the selected parameters, move closer to the front when evaluated with the true

parameters. While this is interesting to see in reality this should not be impactful as they would not have been considered as possible solutions. It has also been noted that points within the central region shift away from the Pareto front. This shows that none of the found points lie within this region meaning that it is likely due to insufficient optimization rather than poor calibration.

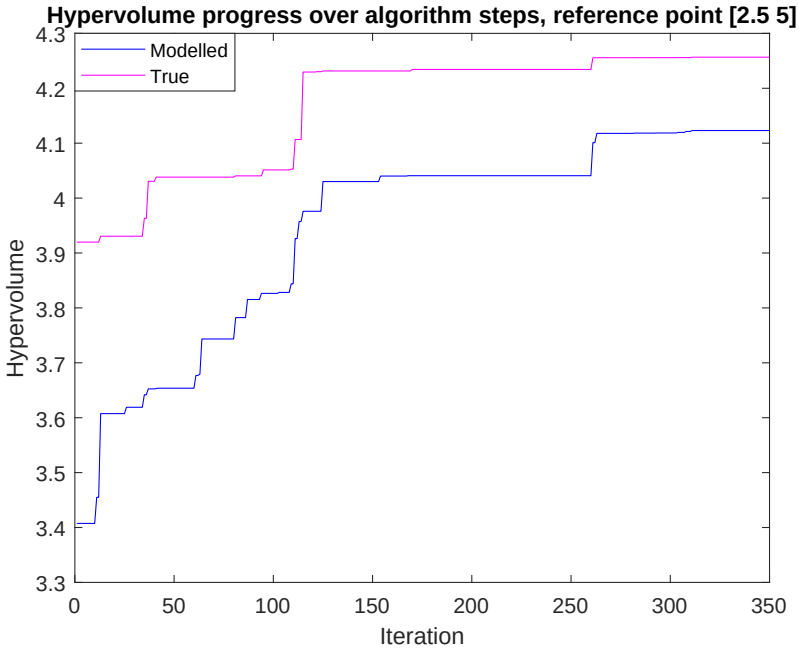


Figure 7.12: Hypervolume progress for both the modelled and true output populations over the cause of the optimization iterations. There is initially a large error present due to parameter error which is reduced over time. The rate of improvement slows as points closer to the front are obtained. An error remains due to the model not identifying points near the central region of the Pareto front.

The performance of the optimization is assessed with the hypervolume indicator using the reference point located at [2.5 5]. The hypervolume for the modelled and true outputs produced using both the new input sets and initial points (seen in Fig. 7.12), it can be seen that they continuously improve over the course of the algorithm. Initially, as expected there is a large difference between the hypervolumes for the modelled and true outputs. This difference is reduced over the course of the algorithm but reaches a point after which only minor reductions in the improvement are obtained. The continued difference between the two hypervolumes is believed to be due in large part to the bad parameter values at which the initial points were evaluated. As they are never re-assed using newly obtained parameters due to the limitation of the evaluation budget it is not possible to realise that they in fact perform relatively well. As more points were discovered closer to the

front the rate of improvement of the hypervolume slowed down. The final hypervolume reached by the true outputs was 4.26 and the final hypervolume reached by the modelled points was 4.09. Realistically the maximum hypervolume possible is 4.83 (determined from taking 100,00 random Pareto optimal samples and calculating the obtained hypervolume) so achieving these hypervolumes is reasonable. Most of the missed hypervolume is present at the extremities which were not fully explored and central portion of the Pareto front where the optimization was unable to locate points close to the front.

7.4.4 Results of running the classical method

This next section details the results generated by the classical approach of calibration followed by optimization. This run uses the same seed as the alternating method and hence generates the same initial population of points, which is done to ensure that variation in the results is due to the differences in the approach and not different initial data. The only difference between the initial information available for the series run (classical approach) is that all ten of the available expert points are evaluated at initialization rather than being incrementally added over the cause of the algorithm.

The results are displayed using the same plots to allow for comparison between the two methods. The progress of the calibration can be seen in Fig. 7.13.

While there are points which lie away from the true value, many of these points can be explained as corresponding with local minima's (at around 0, 0.77 and 1). A large proportion of the remaining selected points that are not near the true value for parameter 1 lie between 0.5 and 0.6. There does not appear to be a reason for this to happen and so is most likely caused by possessing expert points that do not manage to adequately represent the model. The fact that most of the selected values for parameter 2 lie either close to 0.5 or 3 indicates that the true value that would have produced the best results for these points lies outside of the search range. There does not appear to be a trend of the selected values improving over the cause of the calibration although some better values for parameter 2 were spotted towards the end.

As this method is looking at utilising the full calibration budget before moving on to perform optimization only a single parameter set is selected. For this run the selected parameters are [0.501 0.504] which may seem odd as points possessing good approximations of parameter 1 are present. The reason that the points lying closer to the true value were discarded is due to them having worse likelihoods which could have been caused by either the expert population or the inability to select an appropriate value of parameter 2.

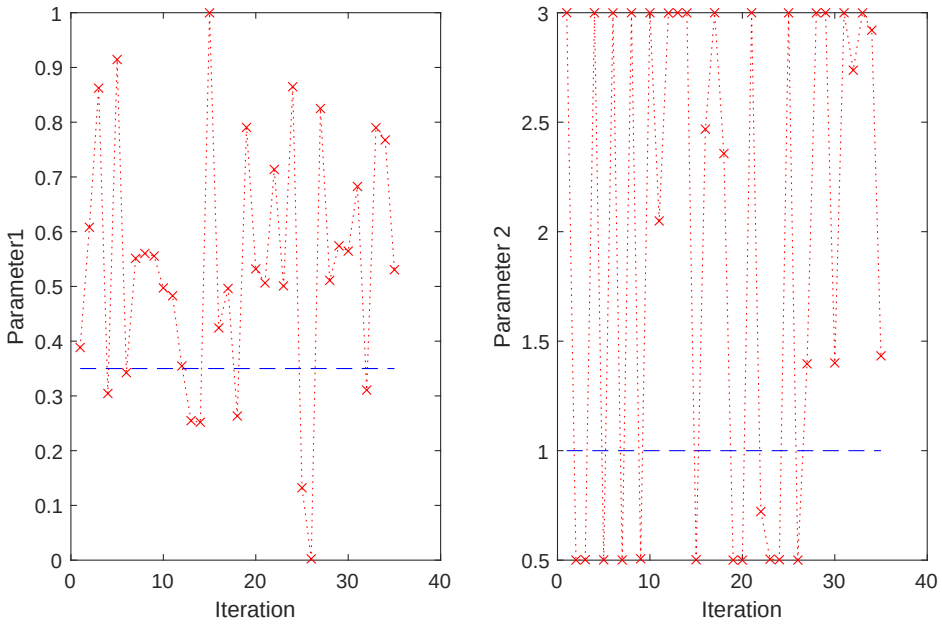


Figure 7.13: Selected parameter value after each interaction of calibration. Dashed line shows the true parameter value, blue x's show initial training data spread. There is no clear trend of improvement observed over the course of the calibration. The parameter set found to be producing the highest likelihood was [0.501 0.504].

The effect of the different parameter sets selected over the cause of the calibration on both the modelled expert inputs as well as points lying on the Pareto front can be observed in Fig. 7.14. All the points except for the additional initial expert points match those examined when looking at the alternating method.

The classical approach exhibits that there is no clear trend of improvement over the 35 calibration iterations (see Fig. 7.14). The error present at points lying on the Pareto front is consistently higher than that present resulting from the outputs of the expert inputs. There are multiple instances at which the error for all points is noticeably reduced. Comparing this to the progress made with calibrating the parameters (see Fig. 7.13), these cases can be seen to have been caused by points for which their value of parameter 1 lies close to the true value. Looking at the parameter set that was selected to be taken forward for use within optimization (iteration 23) it does not arear to be performing well. However, scrutinizing it more closely it can be observed that while the error present at the Pareto front is not improved the error the expert points is in fact greatly diminished.

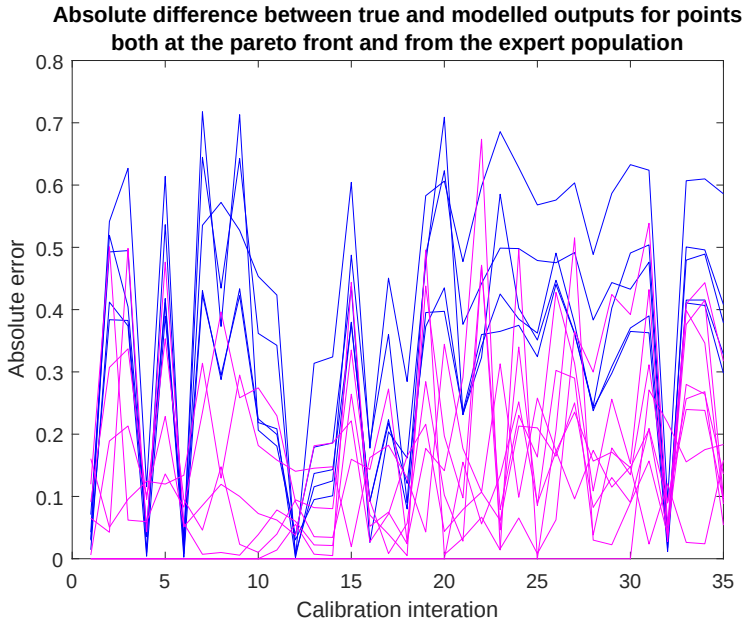


Figure 7.14: The effects of the different parameter sets selected during calibration on both expert and Pareto optimal points. While there are some iterations at which low error is observed, there does not appear to be any continuous improvement over the course of the calibration. In the majority of cases the performance of the expert points is better than that of points chosen on the Pareto front.

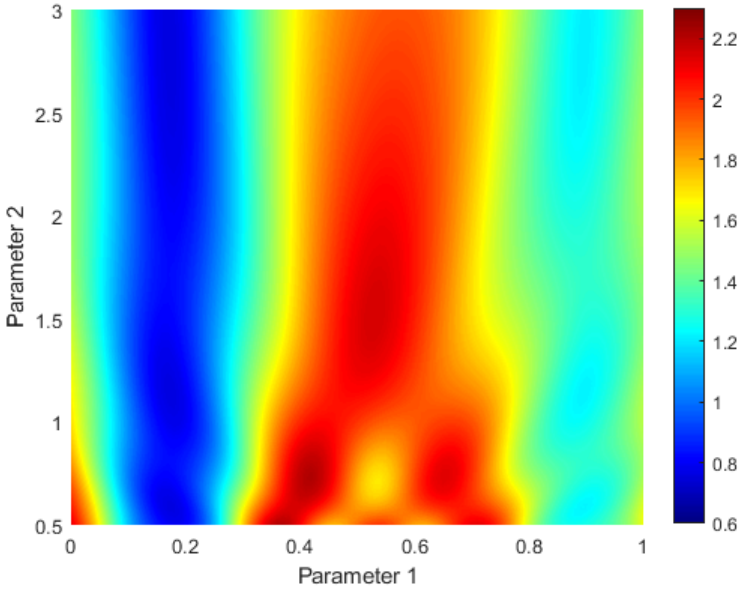


Figure 7.15: Model output 1 for fixed inputs with varying parameters. Included to allow for easier comparison with plots in Fig. 7.16. The model outputs match those presented in Fig. 7.7.

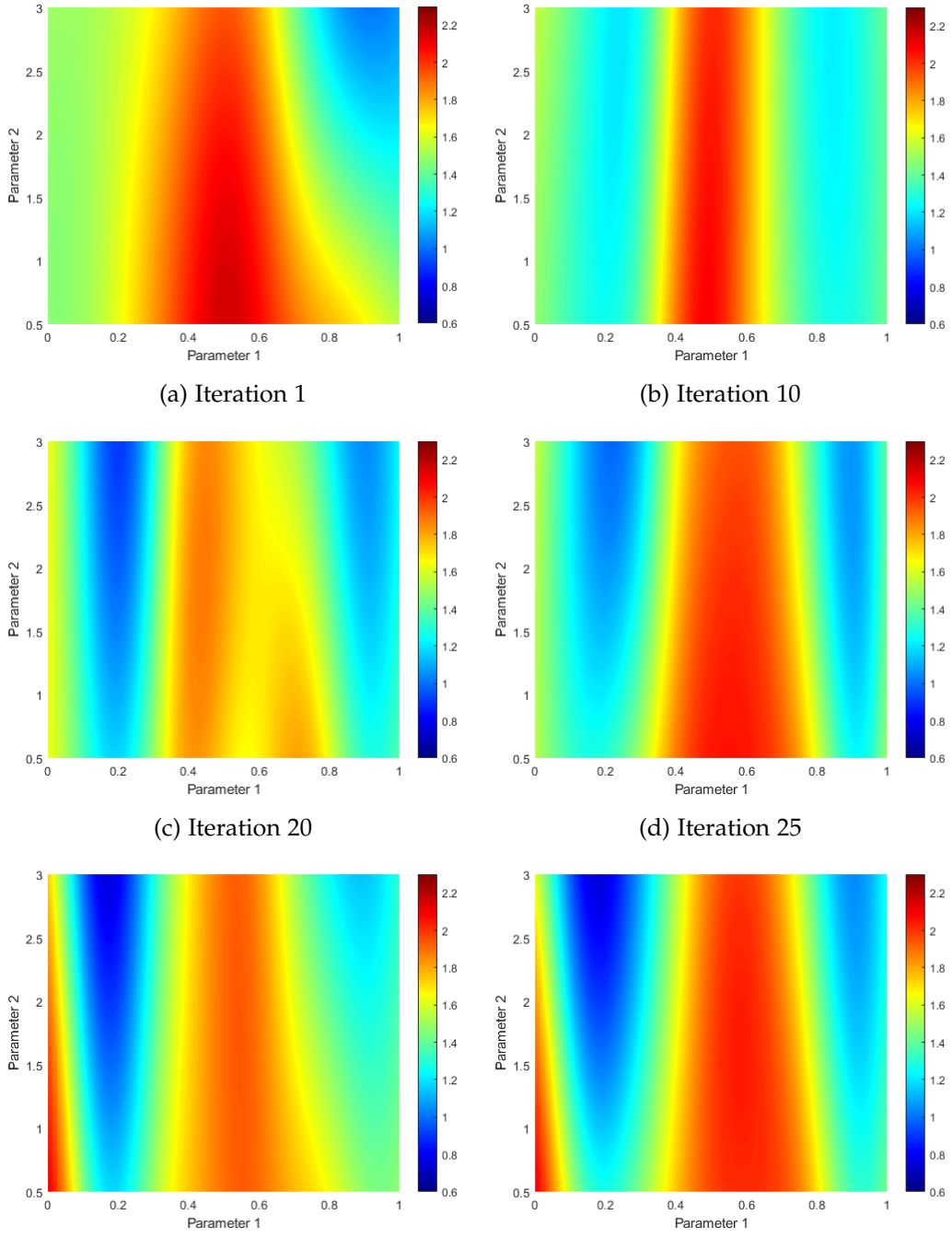
The model output for objective one with varying parameters is shown in Fig. 7.15. Due to using the same test function and initial population the model output possesses the same shape as that seen for the model in the alternating method and has been included mainly to allow for easier comparison to the surrogates presented in Fig. 7.16.

The selected calibration iterations that have been chosen for presentation for the series method, in Fig. 7.16, are different to those that were displayed for the alternating method. The main reasons for this are that the total number of calibrations iterations used is less due to more points being generated at each iteration and so the available budget is used up faster. As additional information is acquired over the course of the calibration additional iterations were selected later to examine if the added information has an impact on the surrogate.

The first stage selected and displayed in Fig. 7.16.a is of the initial surrogate built which as would be expected matches the initial surrogate produced by the alternating method. As seen previously the two main troughs are not currently well modelled and none of the more complex regions are identified. At the tenth iteration, which can be seen in Fig. 7.16.b, the main regions of the surrogate are clear although both their shape and the size of the output values do not match those present within the model well.

The shape of the surrogate matches that of the model by the 20^{th} iteration, seen in Fig. 7.16.c, and is maintained until the completion of the calibration. Over the course of this period small changes in the size and shape of the output surface can be observed. The main difference to the model is that the surrogate has continued to not include the more complex regions that are present for smaller values of parameter 2.

The absolute error between the surrogate and model output plotted in Fig. 7.17 gives a much clearer image of how the surrogate changes. The iterations displayed are again selected to match the surrogates displayed in Figure 7.16. The peak error present at the different iterations of the series method is overall lower than that perceived in the surrogates produced for calibration by the alternating method. The error present due to badly modelling the trench, that occurs at small values of parameter 1, does not get removed over the cause of the calibration. More complex regions of the surrogate's surface are often observed as peaks on the error plot due to failing to be properly modelled by the surrogate. This shows again that either population of points used to build the surrogate failed to include points containing information about the peaks and troughs or the surrogate is struggling to model them.



(a) Iteration 1

(b) Iteration 10

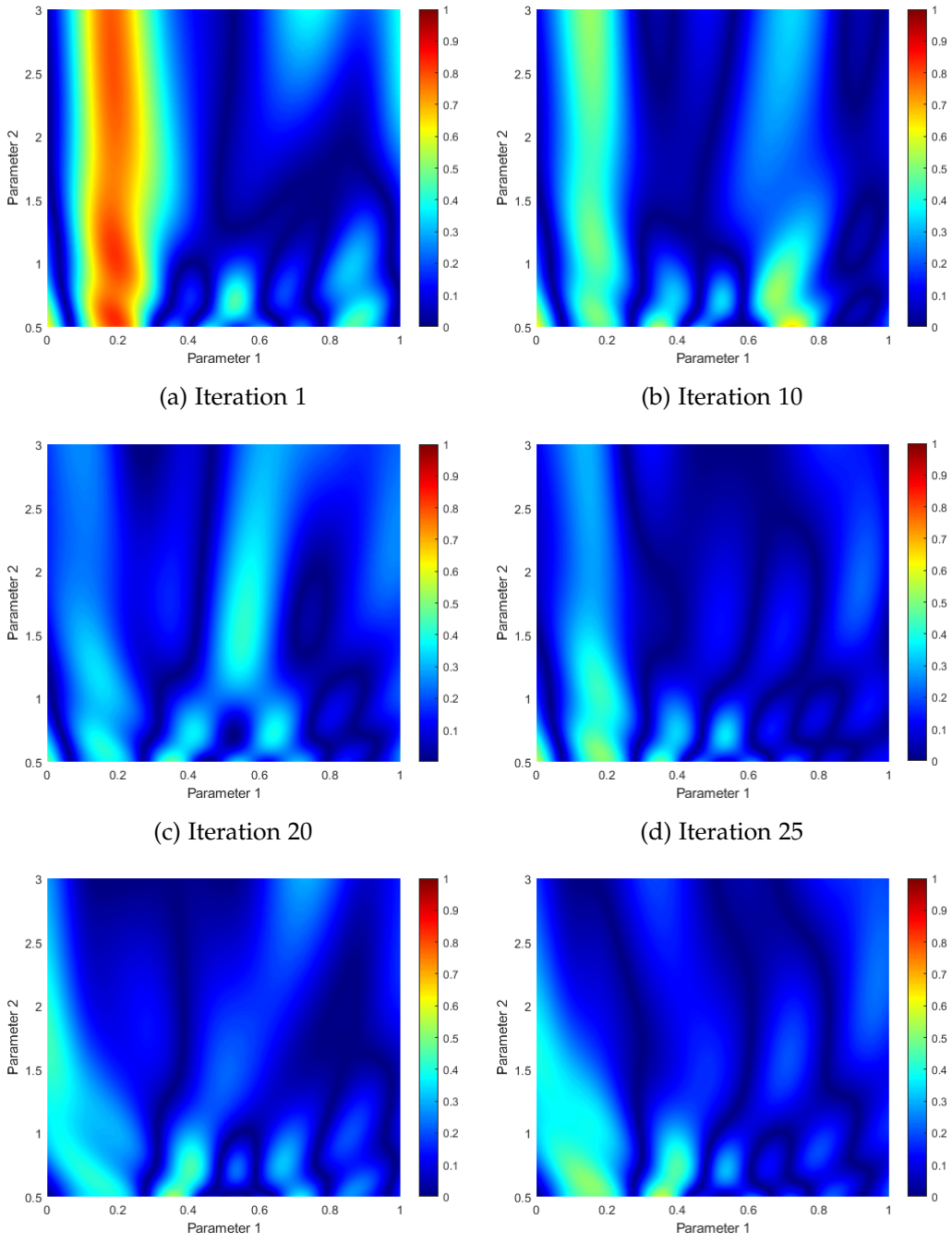
(c) Iteration 20

(d) Iteration 25

(e) Iteration 30

(f) Iteration 35

Figure 7.16: Plots of the surrogate looking at the effects of the parameters on output 1 after 1, 10, 20, 25, 30 and 35 calibration iterations. It appears that the surrogate struggles to correctly identify the channel which is present for lower values of parameter 1. Like before the surrogate failed to identify the more complex regions occurring at lower values of parameter 2.



(a) Iteration 1

(b) Iteration 10

(c) Iteration 20

(d) Iteration 25

(e) Iteration 30

(f) Iteration 35

Figure 7.17: The absolute error between the first surrogate and model output after 1, 10, 20, 25, 30 and 35 calibration iterations. Regions where peaks and troughs occur can be seen to possess a larger absolute error. It can be clearly seen that as noted in Fig. 7.16 the surrogate appears to struggle with correctly identifying the troughs present for smaller values of parameter 1.

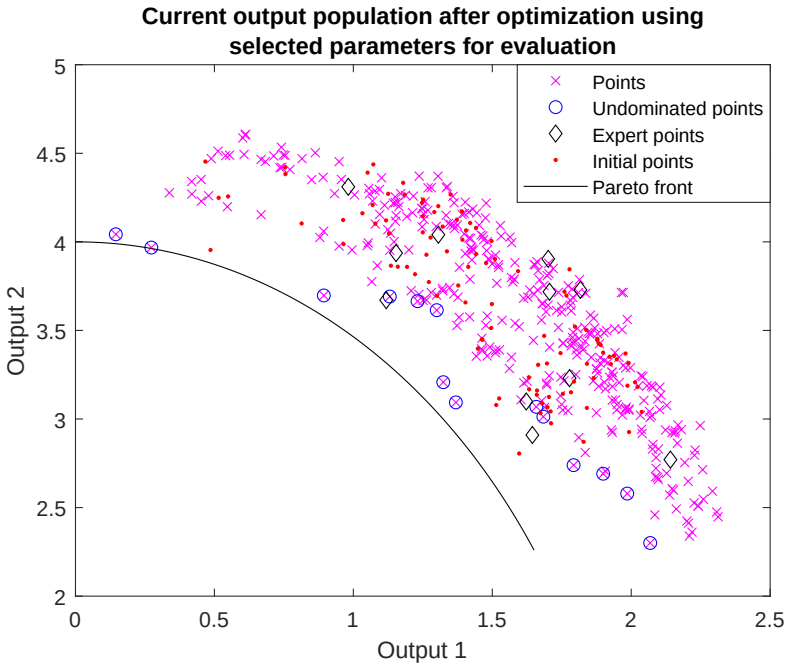


Figure 7.18: The final output population obtained when points were evaluated using the parameters selected from the calibration runs. Only a few points manage to make it close to the true Pareto front. All the selected expert points lie away from the true front. The obtained points appear to be forming a front set back from the true Pareto front.

The results of the optimization shown in Fig. 7.18 appear worse than those produced when using the alternating method. There are a smaller number of points discovered close to the front with those that are found lying further apart. The classical method managed to obtain points located within the central region of the output space which are closer to the true Pareto front than in the alternating method. There expert points present within the classical approach lie further back with none of them being selected close to the Pareto front. The points selected by the optimization form two fronts that are set back from the Pareto front.

When the points discovered over the course of performing optimization are evaluated using the true parameters a large shift in their location can be observed in Fig. 7.19. Many points experienced an increase in the value of their second objective to the extent that no point remain that only one point remains that possesses a value for output 2 which is lower than 2.5. There were multiple points obtained within the central region after evaluation with the true parameters which is an area that both methods appeared to have difficulty in identifying. The Pareto optimal set determined from the points evaluated using selected parameters can be seen to lose out in many cases to points that have shifted after being evalu-

ated with the true parameters. This clearly shows how the incorrect parameters selection has a major impact on the perceived performance of the methodology.

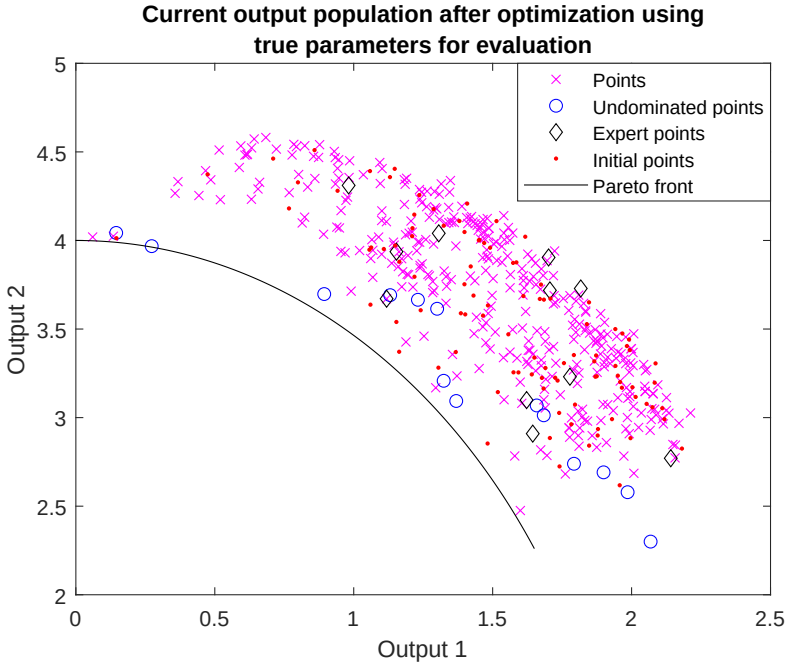


Figure 7.19: The final output population obtained when points were evaluated using the true parameters. Points in the central and lower region of the front have moved closer to the true front. The points obtained with lower values of output 2 have been shifted dramatically when comparing to Fig. 7.18.

The performance of the optimization over algorithm run for the classical method can be seen in Fig. 7.20. Similarly, to the hypervolume observed for the alternating method in Fig. 7.12 the true performance again is much superior to that produced by the model, which as has been stated is due to the initial population. The final modelled hypervolume achieved is 4.09 whereas the final hypervolume gained from the points being evaluated using the true parameters is 4.24. This performance is comparable to that achieved by the alternating method, being only slightly worse.

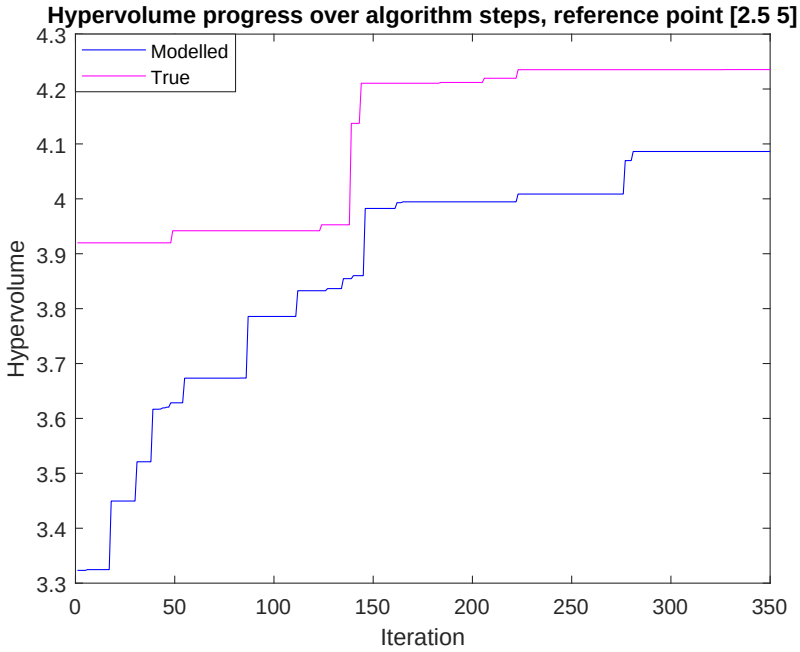


Figure 7.20: Hypervolume progress for both the modelled and true output populations over the cause of the optimization iterations for the classical method. The parameter error produces a large initial difference between the modelled and true hypervolume. The rate of improvement for the modelled hypervolume slows over the cause of the optimization. The modelled hypervolume fails to match the true hypervolume, although the difference between the two is reduced over the cause of the optimization.

7.4.5 Comparison

While the results for the two different methods have been laid out in the previous sections, there are some areas that could benefit from a clearer comparison. The first of these is an examination of the positioning of the expert points which were obtained during both methods.

The inputs for the selected points are displayed in Table 7.4. The first five points are those shared between both methods which were selected at initialization. The remaining five points for the classical method were also selected at initialization through the used of Latin hypercube sampling as described in the methodology. The additional alternating points are listed in the order that they were obtained during the run. The first two inputs depict the direction of the point from the origin while the last two inputs dictate the distance between the point and the Pareto front. For a point to lie on the Pareto front it is necessary for the third and fourth inputs to be 2.1 and 2.8 respectively. From the table the discovered alternating points are closer to these values than those acquired by the

Classical	Alternating
[0.33 3.59 0.91 1.76]	
[1.36 1.49 2.95 0.94]	
[0.98 0.03 4.14 5.00]	
[1.78 3.04 1.38 3.74]	
[0.46 2.34 5.55 6.70]	
[1.70 3.55 2.60 1.82]	[1.58 0.65 3.35 3.78]
[0.45 1.67 5.23 4.11]	[1.29 0.28 2.72 2.40]
[1.28 2.48 4.22 6.54]	[1.75 0.00 1.44 1.61]
[0.97 0.64 0.27 0.96]	[0.75 2.61 2.67 1.26]
[0.01 1.51 2.17 4.99]	[0.85 1.66 1.86 2.99]

Table 7.4: The expert points that were available for the classical and alternating method displayed in the order they were acquired. Those obtained by the alternating method in general have better values (closer to lying on the true Pareto front) than those obtained by the classical method.

classical method most of the time.

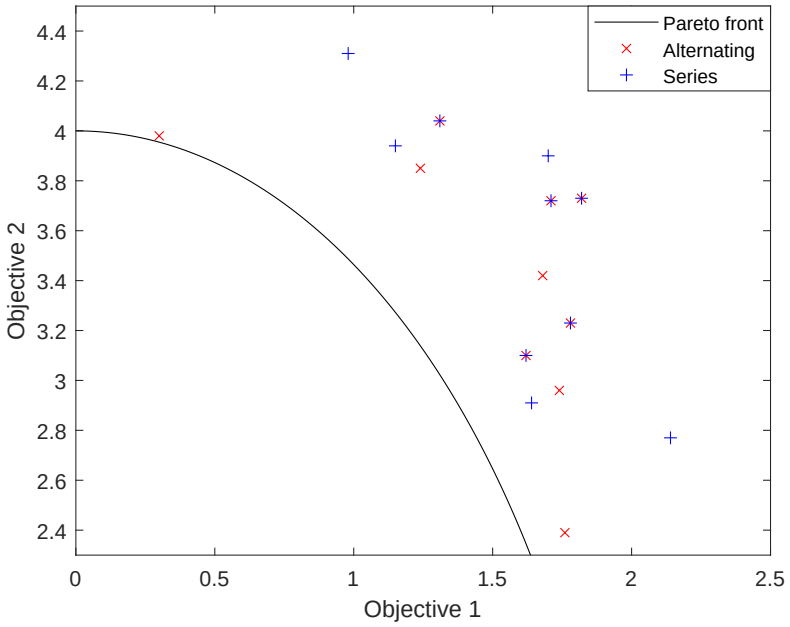


Figure 7.21: Comparison of the expert points present within the alternating and classical (series) methods. The points selected by the alternating method are in general closer to the front than those obtained by the series classical method.

The output resulting from these input sets can be seen displayed in Fig 7.21 along with the true Pareto front. It can be clearly seen that the alternating method successfully obtained points closer to the Pareto front than those determined

through Latin hypercube sampling. The alternating methods expert points that lie on either side are closer to the front, this is explained by the optimizers progress which can be seen in Fig. 7.10. As for the expert points obtained at initialization for the classical method, while they have the potential to be chosen close to the front this is unlikely to happen as was seen here.

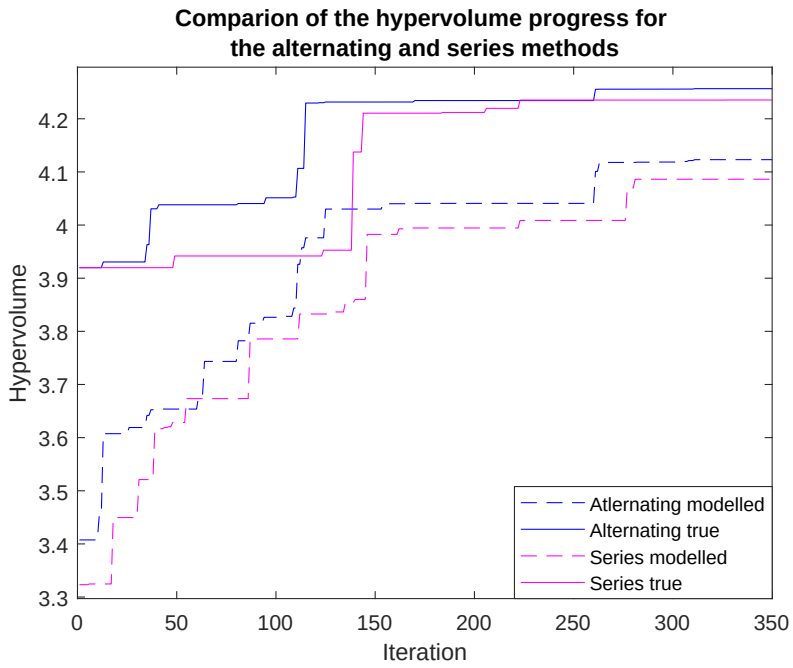


Figure 7.22: Comparison of the hypervolumes achieved by alternating and classical (series) methods. The rate of improvement for the alternating method is faster than the classical method for both the true and modelled values. The final hypervolumes achieved by the alternating method outperformed those by the series method.

While the hypervolumes obtained by both the alternating and classical method have been previously shown they have been plotted here again to allow for an easier comparison. Looking at Fig. 7.22 it can be seen that the alternating method allowed for faster improvement in the output as well as a better final hypervolume. In both cases there is a large discrepancy between the modelled and true output. While from looking at Fig. 7.10 and Fig. 7.11 it can be seen that this error appears to mainly come from the difference in the initial population when it is evaluated with the expert parameters rather than those chosen from the calibrated set. In the case of the classical method it appears as if a lot more of the error between the modelled and true hypervolume was due to movement of optimized points which were badly calibrated, seen in Fig. 7.18 and Fig. 7.19.

7.4.6 Discussion

It is important to note that the results displayed here show the outcome of a single run of both the alternating and classical methods. Both the runs are setup with the same initial random number generator (RNG) seed which means that the initial information in both runs is the same. This allows us to compare the two runs, as any discrepancy between the performance of the runs must be largely impacted by the change of methodology. Due to using random samples in multiple parts of the method it is not possible to ensure that the same selection was made, and so possible discrepancy could have formed due to this.

From examining multiple runs of the methods with differing RNG seeds, and hence initial populations, it was found that the performance of the methods was strongly affected. The results set which is presented comes from a run in which the Latin hypercube sampling performed by the classical method, to find its expert points, does not determine one situated next to the Pareto front. This was chosen as we are interested in how the alternating method can help when the region of interest is not identified initially.

It is necessary to note that in this case, at first glance, the final performance metric value achieved by the alternating and classical methods may appear very close. While this may be true there are other aspects that need to be considered; such as the density of points near the Pareto front and the calibration performance. It was seen that the calibration achieved through the alternating method was superior to that of the classical approach. Over the course of the run the alternating method managed to identify a parameter set which was close to the true parameters. It also focused on improving the performance for both the expert population as well as points lying near the true Pareto front.

Another aspect of discussion is the surrogate's ability to correctly model the true function. The kriging model ended up struggling to model the areas of higher complexity within the output space. Even with this issue the alternating method managed to identify reasonable approximation of the true parameter values. The examination of other possible surrogate methods is one area which could be worth looking into in the future.

7.5 Conclusion

This chapter has detailed an extension to the alternating method designed to allow it to function with the use of a smaller budget of evaluations. This has been done through the addition of a kriging surrogate model that is used within both the model calibration and optimization. Both this new alternating method as well as

a classical alternative are detailed along with justification forwhy certain design choices were taken.

Before the results of running either of the setups are presented a small investigation into the number of generations required to find a suitable θ value for use within the surrogate was carried out. It was concluded that 25 generations were required before the marginal return became insubstantial.

From the results presented within this chapter, it was found that the alternating method achieved superior calibration to the classical approach. Additionally, the alternating method was found to improve the performance of both points lying within the expert population and those near the Pareto front. In the future it is necessary to perform further work into assessing different surrogate model as the current kriging model was shown to have issues.

Chapter 8

Conclusion

The principle aim of this research was to determine if there was a method by which the development steps of model calibration and optimization could be made more efficient through being considered as a single combined problem. This thesis has laid out a combined mathematical formulation for the joint problem of model calibration and optimization (Chapter 4). It presents methods by which such problems can be benchmarked along with a set of potential test problems which include both calibration parameters and control inputs for identification (Chapter 5). An initial examination of the use of an alternating method for improving the performance of the joint problem over the classical serial approach was performed (Chapter 6). This was further expanded to look at cases in which expensive multi objective problems were being considered (Chapter 7). With the reduction in the available evaluation budget a new surrogate method was developed and its performance assessed.

8.1 Main contributions

The key original knowledge contributions towards the fields of model calibration and optimization presented within this thesis are:

- The construction of a new mathematical formulation to express the combined problem of model calibration and optimization. While the different components of the formulation are based on those drawn from the optimization and calibration literature, the new framework provides consistent notation allowing for a coherent expression of the combined problem. This new framework allows for better tracking of components, such as uncertainty, throughout the entire process.

- The extension of pre-existing benchmark problems to create new variants which would allow for testing of methods designed to solve the combined stages of model calibration and optimization. This was accomplished through selecting popular test problems from within the optimization literature and expanding them to incorporate both model parameters along with control inputs. The multi-objective problems were also expanded to incorporate cases where both modelling error is present as well as absent.

- A new component for the WFG framework, *s_signal*, that incorporates model parameters was developed. The new component was designed so that it would possess adjustable complexity and could not be easily approximated via a simple surrogate model. This was necessary due to the way many of the current test problems are formed by adding complexity on to simpler

problems. The new module for WFG is also designed to be non-separable and allow for scalability.

- The proposition of a new alternating methodology for solving the combined problems of model calibration and optimization. This new method moves between the two stages with the aim of making more efficient use of the available evaluation budget. The alternating method was developed under the assumption that successfully calibrating the model for a given set of inputs did not guarantee that the model would be calibrated for all inputs. Through alternating between the two stages, the aim was to obtain better knowledge of more relevant area of the output space and hence ensure that the outputs obtained lie as close as possible to the true front.

- Extension of the alternating methodology for use with a small evaluation budget. This was achieved through the incorporation of surrogate models within both the calibration and optimization stages. These surrogates allowed for cheap evaluations to be performed with the aim of aiding the solver to determine the optimal locations to use the available model evaluations. The surrogates also facilitate better information sharing between the two stages.

- Assessments for the performance of both the new alternating method as well as its surrogate version were performed. In both cases they are compared to a comparable setup in which the classical approach, of performing the two stages in series, was implemented. The basic alternating method was examined on the DTLZ1$_\theta$ function as well as ZDT1$_\theta$ and WFG2$_\theta$ for the cases when model error was present and absent. It was shown that the alternating method aided in improving the final parameters achieved by calibration. It was also demonstrated that the alternating method to achieve comparable (in some cases superior) performance on average for the final hypervolume. The extended alternating method was examined on the newly developed WFG4$_s$ function. It again showed an improvement in the calibration as well as achieving faster progress within the found hypervolume.

8.2 Future work

At the current stage there are a number of directions future research and improvements, based on the work presented, could be carried out for.

One area of future work to consider is the continued development of the alternating approach. Some of its components, which could benefit from further

examining, include the effects of using different conditions to determine when the method would switch between stages. If a dynamic method were used to determine the switching time it would be necessary to consider the conditions under which the next true evaluation would take place. Another component which may be beneficial to analyse is how the use of different surrogate models affects the performance. Further testing of the current setups on a larger variety of problems may aid in achieving a better understanding of how the new approach operates.

Another avenue of future work should be the implementation of a robust methodology using implicit averaging, to allow it to function with the limited evaluation budget, as a viable alternative for comparison to the alternating method. The possibility of incorporating the robustness indicator into the alternating approach should also be considered. This could be done either as a replacement of the current optimization or as a supporting mechanism. One potential benefit of doing this would be an ability to help mitigate any issues from cases in which the model calibration has failed to successfully calibrate the parameters.

The final area of future work is to apply the new method to a real-world case study. Currently the work carried out has all been based on test functions which, while allowing for some performance examination, do not necessarily represents how the method would truly function.

Bibliography

J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation*, 19(1):45–76, 2011.

J.M. Bader. *Hypervolume-based search for multiobjective optimization: theory and methods*. Number 112. Johannes Bader, 2010.

T. Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418, 1763.

T. Bäck and H.P. Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1):1–23, March 1993.

P. Beerli. Comparison of Bayesian and maximum-likelihood inference of population genetic parameters. *Bioinformatics*, 22(3):341–345, February 2006.

R. Bellaan. The theory of dynamic programming. Technical report, Rand corp santa monica ca, 1954.

N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181 (3):1653–1669, September 2007.

H.G. Beyer and B. Sendhoff. Robust optimization – a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33):3190–3218, July 2007.

L.C.T. Bezerra, M. López-Ibáñez, and T. Stützle. Comparing decomposition-based and automatically component-wise designed multi-objective evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization, EMO 2015 Part I*, pages 396–410, 2015.

J. Bongard, V. Zykov, and H. Lipson. Resilient Machines Through Continuous Self-Modeling. *Science*, 314(5802):1118–1121, November 2006.

S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, UK ; New York, 2004.

K. Bringmann, T. Friedrich, and P. Klitzke. Two-dimensional subset selection for hypervolume and epsilon-indicator. pages 589–596. 2014.

D.S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

L.P. Cáceres, M. López-Ibánez, and T. Stützle. Ant colony optimization on a budget of 1000. In *International Conference on Swarm Intelligence*, pages 50–61. 2014.

L.P. Cáceres, M. López-Ibáñez, and T. Stützle. Ant colony optimization on a limited budget of evaluations. *Swarm Intelligence*, 9(2-3):103–124, September 2015.

T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, 23(9):3137–3166, May 2019.

G. Damblin, M. Couplet, and B. Iooss. Numerical studies of space-filling designs: optimization of Latin Hypercube Samples and subprojection properties. *Journal of Simulation*, 7(4):276–289, November 2013.

J. Davins-Valldaura, S. Moussaoui, G. Pita-Gil, and F. Plestan. ParEGO extensions for multi-objective optimization of expensive evaluation functions. *Journal of Global Optimization*, 67(1-2):79–96, January 2017.

K. Dächert, J. Gorski, and K. Klamroth. An augmented weighted Tchebycheff method with adaptively chosen parameters for discrete bicriteria optimization problems. *Computers & Operations Research*, 39(12):2929–2943, December 2012.

K. Deb et al. *Multi objective optimization using evolutionary algorithms*. John Wiley and Sons, 2001.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2): 182–197, 2002.

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145, 2005.

J.A. Duro, R.C. Purshouse, S. Salomon, D.C. Oara, V. Kadirkamanathan, and P.J. Fleming. sparego – a hybrid optimization algorithm for expensive uncertain multi-objective optimization problems. In K. Deb, E. Goodman, C.A.

Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, editors, *Evolutionary Multi-Criterion Optimization*, pages 424–438, Cham, 2019.

M. Ehrgott, J. Ide, and A. Schöbel. Minmax robustness for multi-objective optimization problems. *European Journal of Operational Research*, 239(1):17–31, November 2014.

L. Excoffier and M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, September 1995.

A.I. Forrester and A.J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1-3):50–79, January 2009.

H. Fu, P.R. Lewis, B. Sendhoff, K. Tang, and X. Yao. What are dynamic optimization problems? In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1550–1557. 2014.

I. Giagkiozis, R.C. Purshouse, and P.J. Fleming. An overview of population-based algorithms for multi-objective optimisation. *International Journal of Systems Science*, 46(9):1572–1599, 2015.

M.S. Gibbs, G.C. Dandy, and H.R. Maier. Calibration and Optimization of the Pumping and Disinfection of a Real Water Supply System. *Journal of Water Resources Planning and Management*, 136(4):493–501, July 2010.

J. Goodman and J. Weare. Ensemble samplers with affine invariance. *Communications in applied mathematics and computational science*, 5(1):65–80, 2010.

H.M. Gutmann. A radial basis function method for global optimization. *Journal of global optimization*, 19(3):201–227, 2001.

A. Habib, H.K. Singh, and T. Ray. A multi-objective batch infill strategy for efficient global optimization. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 4336–4343. 2016a.

A. Habib, H.K. Singh, and T. Ray. A study on the effectiveness of constraint handling schemes within Efficient Global Optimization framework. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, pages 1–8. 2016b.

J. Hakanen and J.D. Knowles. On Using Decision Maker Preferences with ParEGO. In H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M. Wiecek,

Y. Jin, and C. Grimme, editors, *Evolutionary Multi-Criterion Optimization*, volume 10173, pages 282–297. Springer International Publishing, Cham, 2017. DOI: 10.1007/978-3-319-54157-0_20.

N. Hansen, S. Finck, and R. Ros. *Coco-Comparing continuous optimizers: The documentation*. PhD thesis, INRIA, 2011.

W.K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97, April 1970.

D. Horn, T. Wagner, D. Biermann, C. Weihs, and B. Bischl. Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark. In A. Gaspar-Cunha, C. Henggeler Antunes, and C.C. Coello, editors, *Evolutionary Multi-Criterion Optimization*, volume 9018, pages 64–78. Springer International Publishing, Cham, 2015. DOI: 10.1007/978-3-319-15934-8_5.

D. Huang, T.T. Allen, W.I. Notz, and N. Zeng. Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. *Journal of Global Optimization*, 34(3):441–466, March 2006.

S. Huband, L. Barone, L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 280–295. Springer, 2005.

E.J. Hughes. MSOPS-II: A general-purpose Many-Objective optimiser. pages 3944–3951. September 2007.

H. Ishibuchi, T. Doi, and Y. Nojima. Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms. *Parallel Problem Solving from Nature-PPSN IX*, pages 493–502, 2006.

H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima. Simultaneous use of different scalarizing functions in MOEA/D. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 519–526, 2010.

H. Ishibuchi, N. Akedo, and Y. Nojima. A study on the specification of a scalarizing function in MOEA/D for many-objective knapsack problems. In *International Conference on Learning and Intelligent Optimization*, pages 231–246, 2013.

H. Ishibuchi, H. Masuda, and Y. Nojima. A Study on Performance Evaluation Ability of a Modified Inverted Generational Distance Indicator. pages 695–702. 2015.

H. Ishibuchi, H. Masuda, and Y. Nojima. Sensitivity of performance evaluation results by inverted generational distance to reference points. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1107–1114. 2016.

H. Ishibuchi, K. Doi, and Y. Nojima. On the effect of normalization in MOEA/D for multi-objective and many-objective optimization. *Complex & Intelligent Systems*, 3(4):279–294, 2017.

E.T. Jaynes. *Bayesian methods: General background*. Cambridge University Press, 1 edition, November 1986.

Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

O. Jones, R.C. Purshouse, and J.E. Oakley. Toward a unified framework for model calibration and optimization in virtual engineering workflows'. October 2019.

O.T. Kajero, R.B. Thorpe, T. Chen, B. Wang, and Y. Yao. Kriging meta-model assisted calibration of computational fluid dynamics models. *AIChE Journal*, 62 (12):4308–4320, December 2016.

M.I. Kamien and N.L. Schwartz. *Dynamic optimization: the calculus of variations and optimal control in economics and management*. Courier Corporation, 2012.

M.C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.

J.P.C. Kleijnen, W. van Beers, and I. van Nieuwenhuyse. Expected improvement in efficient global optimization through bootstrapped kriging. *Journal of Global Optimization*, 54(1):59–73, September 2012.

J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, February 2006.

J. Knowles and E.J. Hughes. Multiobjective optimization on a budget of 250 evaluations. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 176–190. 2005.

J. Knowles, D. Corne, and A. Reynolds. Noisy multiobjective optimization on a budget of 250 evaluations. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 36–50. 2009.

R. Kwiatkowski and H. Lipson. Task-agnostic self-modeling machines. *Science Robotics*, 4(26):eaau9354, January 2019.

R.S. Langley. Unified approach to probabilistic and possibilistic analysis of uncertain systems. *Journal of engineering mechanics*, 126(11):1163–1172, 2000.

M. Laumanns, E. Zitzler, and L. Thiele. On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 181–196, 2001.

L. Lebensztajn, C. RondiniMarretto, M. CaldoraCosta, and J.L. Coulomb. Kriging: A Useful Tool for Electromagnetic Device Optimization. *IEEE Transactions on Magnetics*, 40(2):1196–1199, March 2004.

X. Li, D.E. Weller, and T.E. Jordan. Watershed model calibration using multi-objective optimization and multi-site averaging. *Journal of Hydrology*, 380(3-4): 277–288, January 2010.

K.H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge Based Intelligent Engineering Systems*, 4(3):172–183, 2000.

J.S. Liu. *Monte Carlo strategies in scientific computing.* New York, NY: Springer, 2001.

M. Mavrovouniotis, C. Li, and S. Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33:1–17, April 2017.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, pages 1–7, 2004.

J. Müller. MISO: mixed-integer surrogate optimization framework. *Optimization and Engineering*, 17(1):177–203, March 2016.

M.D. Morris and T.J. Mitchell. Exploratory designs for computational experiments. *Journal of statistical planning and inference*, 43(3):381–402, 1995.

Z.P. Mourelatos and J. Liang. A methodology for trading-off performance and robustness under uncertainty. *Journal of Mechanical Design*, 128(4):856, 2006.

T. Murata and H. Ishibuchi. MOGA: Multi-objective genetic algorithms. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 1, page 289. 1995.

I.J. Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1):90–100, February 2003.

T.T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, October 2012.

W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted$\backslash$ mathcal ${$s$}$-Metric selection. In *International Conference on Parallel Problem Solving from Nature*, pages 784–794. 2008.

R.C. Purshouse and P.J. Fleming. Why use elitism and sharing in a multi-objective genetic algorithm? In *Proceedings of the 2002 Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 520–527, 2002.

Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu. MOEA/D with adaptive weight adjustment. *Evolutionary Computation*, 22(2):231–264, 2014.

H. Qiang, Y. Fuyuan, Z. Ming, and O. Minggao. Study on Modeling Method for Common Rail Diesel Engine Calibration and Optimization. pages 2004–01–0426, March 2004.

Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6): 712–731, December 2007.

F. Rambeaux, F. Hamelin, and D. Sauter. Optimal thresholding for robust fault detection of uncertain systems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 10(14):1155–1173, 2000.

C.E. Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

N. Riquelme, C. Von Lücken, and B. Baran. Performance metrics in multi-objective optimization. In *Computing Conference (CLEI), 2015 Latin American*, pages 1–11. 2015.

R.J. Rossi. *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*, June 2017. arXiv: 1609.04747.

J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

M.J. Sasena, P. Papalambros, and P. Goovaerts. Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization. *Engineering Optimization*, 34 (3):263–278, January 2002.

J.D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proceedings of the First Int. Conference on Genetic Algortihms*, pages 93–100, 1985.

M.R. Sierra and C.A.C. Coello. A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms. *Technical Report of CINVESTAV-IPN*, 2004.

H.L. Southall and T.H. O'Donnell. Antenna Design Using the Efficient Global Optimization (EGO) Algorithm. Technical report, DTIC Document, 2011.

R.E. Steuer and E.U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical programming*, 26(3):326–344, 1983.

M. Sunnåker, A.G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz. Approximate Bayesian Computation. *PLoS Computational Biology*, 9(1):e1002803, January 2013.

H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M. Wiecek, Y. Jin, and C. Grimme, editors. *Evolutionary Multi-Criterion Optimization*, volume 10173 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham, 2017. DOI: 10.1007/978-3-319-54157-0.

D.A. Van Veldhuizen and G.B. Lamont. Evolutionary computation and convergence to a pareto front. In *Late breaking papers at the genetic programming 1998 conference*, pages 221–228, 1998.

M.G. Villarreal-Marroquín, P.H. Chen, R. Mulyana, T.J. Santner, A.M. Dean, and J.M. Castro. Multiobjective optimization of injection molding using a calibrated predictor based on physical and simulated data. *Polymer Engineering & Science*, 57(3):248–257, March 2017.

S. Voß. Meta-heuristics: The State of the Art. In G. Goos, J. Hartmanis, J. van Leeuwen, and A. Nareyek, editors, *Local Search for Planning and Scheduling*, volume 2148, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. DOI: 10.1007/3-540-45612-0_1.

K.K. Vu, C. D'Ambrosio, Y. Hamadi, and L. Liberti. Surrogate-based methods for black-box optimization. *International Transactions in Operational Research*, 24(3): 393–424, May 2017.

R.K.W. Wong, C.B. Storlie, and T.C.M. Lee. A Frequentist Approach to Computer Model Calibration. *arXiv:1411.4723 [stat]*, September 2015. arXiv: 1411.4723.

H.P. Wynn and R.A. Bates. Emulator technology in engineering design. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 213(3):305–309, 1999.

D. Zhan, J. Qian, and Y. Cheng. Balancing global and local search in parallel efficient global optimization algorithms. *Journal of Global Optimization*, 67(4): 873–892, April 2017.

Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. Technical report, University of Essex, 2009.

Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by MOEA/D with Gaussian Process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2010.

Z. Zhang, T. Wagener, P. Reed, and R. Bhushan. Reducing uncertainty in predictions in ungauged basins by combining hydrologic indices regionalization and multiobjective optimization: PREDICTIONS IN UNGAUGED BASINS. *Water Resources Research*, 44(12), December 2008.

S.Z. Zhao, P.N. Suganthan, and Q. Zhang. Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes. *IEEE Transactions on Evolutionary Computation*, 16(3):442–446, 2012.

E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature*, pages 832–842. 2004.

E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132, 2002.

E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Evolutionary multi-criterion optimization*, pages 862–876. 2007.