Gaussian Process-based Optimization using Mutual Information for Computer Experiments. Application to Storm Surge extremes.

Theodoros Mathikolonis

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Statistical Science

University College London

March 12, 2020

Declaration

I, Theodoros Mathikolonis, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Chapter 5: The results from the local sensitivity analysis of the storm surge extrema and how each physical parameter affects the maximum run-up and bore height was analyzed by Dr Volker Roeber (Universit de Pau et des Pays de l'Adour), who is an expert in the field of coastal engineering and wave modelling.

Signature

to Mum & Dad

Να το φωνάξω τόσο δυνατά που να μην ξανακοιμηθεί κανένα όνειρο στον κόσμο καμιά ελπίδα πια να μην πεθάνει.

Αυτό το αστέρι είναι για όλους μας
Τάσος Λειβαδίτης

I want to shout it so loud that no dream in the world will sleep again and no hope will die any longer.

This star is for all of us
Tasos Leivaditis

Translated by N.N.Trakakis

Abstract

The computational burden of running a complex computer model can make optimization impractical. Gaussian Processes (GPs) are statistical surrogates (also known as emulators) that alleviate this issue since they cheaply replace the computer model. As a result, the exploration vs. exploitation trade-off strategy can be accelerated by building a GP surrogate. In the current study, we propose a new surrogate-based optimization scheme that minimizes the number of evaluations of the computationally expensive function. Taking advantage of parallelism of the evaluation of the unknown function, the uncertain regions are explored simultaneously, and a batch of input points is chosen using Mutual Information for Computer Experiments (MICE), a sequential design algorithm which maximizes the information theoretic Mutual Information over the input space. The computational efficiency of interweaving the optimization scheme with MICE (optim-MICE) is examined and demonstrated on test functions. Optim-MICE is compared with state-of-the-art heuristics. We demonstrate that optim-MICE outperforms the alternative schemes on a large range of computational experiments. The proposed algorithm is also employed to study the extrema of coastal storm waves, such as the ones that observed during Typhoon Haiyan (2013, Philippines). A stretch of coral reef near the coast, which was expected to protect the coastal communities, actually amplified the waves. The propagation and breaking process of such large nearshore waves can be successfully captured by a phase-resolving wave model. However, the computational complexity of the simulator makes optimization tasks impractical. The optim-MICE algorithm is therefore used to find the maximum breaking wave (bore) height and the maximum run-up. In two idealised settings, we efficiently identify

the conditions that create the largest storm waves at the coast using a minimal number of simulations. This is the first surrogate-based optimization of storm waves and it opens the door to previously inconceivable coastal risk assessments.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Serge Guillas for his endless support, invaluable advice and guidance throughout my research, but most importantly for believing in me and giving me the opportunity to start this amazing journey. His passion, expertise, innovative ideas and amazing personality helped me to overcome any obstacles and stay motivated throughout these years.

I am also grateful to Dr Volker Roeber for sharing his extensive knowledge for storm surges and coastal engineering, and his valuable contribution in my thesis.

I would also like to thank UCL and EPSRC (Engineering and Physical Sciences Research Council) for the financial support during my PhD studies. This gave me the opportunity to attend workshops and conferences around the world, share knowledge with other researchers and communicate the impact of my thesis.

My PhD journey wouldn't be the same without the friendly and supportive environment of the academics, administrative staff and fellow PhD students of the Department of Statistical Science. Thanks everyone for making this journey enjoyable and fulfilling.

Thanks to my beloved friends in London and Cyprus for their understanding, support and for all those wonderful times that we all shared together the past four years.

I am eternally grateful to my parents and sisters for their unconditional love and encouragement. They have always supported me and celebrated with me in successes. This journey would not have been possible without you.

The biggest thank is for my partner Sandy who has been by my side throughout this PhD, living every single minute of it, for her unwavering love, support and careness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Let us consider the problem of maximizing a search engine's profits by displaying appealing advertisements that attract user clicks, but we do not know which advertisements have a high likelihood of being clicked. Or when we want to build a sensor network for monitoring a complex spatiotemporal phenomenon, e.g. the temperature in a building, and an optimal placement of a set of sensors is required. Or when we want to build a seawall to protect a coastal area from extreme environmental phenomena such as a storm surge, but what causes an extreme coastal storm and what is the maximum storm wave height that could possibly struck local level are both unknown.

In such problems, a computer model is often built in order to gain information about the problem and describe the entire situation. The more complex the situation, the more complex the computer model built is. Choosing the advertisements that yield the maximum profits or building an optimal sensor network or even finding the maximum storm wave height require us to optimize the computer model. Taking the traditional mathematical approach (e.g. using gradients) is often impossible due to the complexity of the computer model. An alternative option is to start displaying different advertisements and performing various experiments, until we maximize our profits, or making observations about the storm wave height by exploring different combinations of the sources that possibly produce the worst case

scenario. In practice, such activities are typically costly, require the user's attention and patience and most importantly, are time-consuming.

What the maximum is and under what conditions the objective is achieved are the ultimate questions in all these problems. However, the key common question is which observations should we select with the aim to learn as much as possible about the situation and, potentially, achieve the objective in an efficient manner; by keeping the computational resources to the minimum and ensuring the accuracy of the results.

1.1 Impact statement of the thesis

The aim of this study is to maximize a computer model, which is often seen as an unknown function, with the least number of evaluations. The main contributions of the thesis are summarised below:

- We introduce optim-MICE, a new surrogate-based optimization scheme that maximizes a complex and a computational expensive function with the lowest possible number of function evaluations. To overcome the computational burden of running a computer model, the GP regression is used as a statistical surrogate model, which has been shown to be a good approximation of the computer model. The optim-MICE algorithm is built based on two settings in order to determine the input points at which the computer model is evaluated at: the bandit setting, where a balance between exploration and exploitation needs to be achieved, and the parallel setting, where the knowledge about the unknown function is growing rapidly as a lot of information is collected at the same iteration. Taking advantage of the parallel evaluations of the unknown function, the uncertain regions are explored simultaneously, and a batch of input points is chosen using Mutual Information for Computer Experiments (MICE), a sequential design algorithm which maximizes the information theoretic Mutual Information over the input space.

The computational efficiency of interweaving the optimization scheme with MICE is examined and demonstrated on different tests functions. The optim-MICE algorithm is compared with state-of-the-art heuristics in different computational experiments. In addition to the development of the optimization algorithm and the validation of its efficiency, we examine the effects of the main algorithm settings on the overall performance and whether the computational efficiency of optim-MICE is improved. We also consider the possibility of scaling the objective function and whether by stretching or shrinking the uncertain region of the true optimum, a solution close to the true optimum can be found in less computational time.

- The newly developed optimization scheme proposed in the current work is used to study storm wave extrema, such as the ones that recorded during Typhoon Haiyan in Philippines on November 2013. Typhoon Haiyan damaged USD billions of worth of agriculture and infrastructures and completely destroyed Hernani, a small coastal town. The particular event was successfully simulated by BOSZ (Roeber & Bricker, 2015), a computer model built to reproduce the tsunami-like wave that struck the town. However, studies of storm surge, such as optimization or sensitivity analysis which require a large number of evaluations, have not been done yet due to the computational complexity of the computer model.

  Taking advantage of the computational efficiency of the proposed algorithm on several computational experiments, optim-MICE is used to optimise BOSZ and study different aspects of storm wave extrema: the maximum bore height and the maximum run-up. Since both measures are governed by different processes, the computer model needs to be optimized separately, therefore optim-MICE is used twice. The aim of this study is to identify the conditions that could possibly create extreme wave run-up and bore heights. To better understand the different sources of uncertainties and which of them

control the wave run-up and bore height, a local sensitivity analysis around the maximum run-up and the maximum bore height is performed. Furthermore, the input parameters are all combined in a non-dimensional measure to model their effects on the maximum wave run-up.

This is the first surrogate-based optimization of storm waves and it opens the door to previously inconceivable coastal risk assessments.

1.2 Outline of the thesis

The current thesis is organized as follow:

Chapter 2 provides a comprehensive background on the major components incorporated in the proposed optimization algorithm. It includes relevant terminology and concepts as well as key related work available in the literature.

Chapter 3 presents optim-MICE, the novel sequential-based optimization algorithm proposed in the current work that aims to optimize an unknown, complex and computationally expensive function with the lowest possible function evaluations. In this chapter, the optim-MICE algorithm and the entire parallel optimization procedure are described in details. We compare the proposed optimization algorithm with other state-of-the-art heuristics in different computational experiments. Based on the results of the comparison we discuss the algorithm's computational efficiency.

Chapter 4 focus on the main algorithm settings and discuss their effect on the overall performance of the optim-MICE algorithm. A sensitivity analysis is performed on different computational experiments. The opportunity to improve the computational efficiency of the algorithm is also discussed by examining the possibility of scaling the uncertain region.

Chapter 5 shows the effectiveness of the proposed algorithm on a real-world problem. The optim-MICE algorithm is used to study coastal storm waves heights and run-ups, such as the ones that destroyed the town of Hernani during Typhoon Haiyan. The focus of this chapter is to understand the different sources of uncertainties and efficiently identify the conditions that create the largest storm waves at the coast.

Chapter 6 gives a summary of the main results and considers some related open topics for further work.

1.3 Activities related to the thesis

The newly developed optimization scheme proposed in this thesis has been presented at the:

- Statistical and Applied Mathematical Sciences Institute (SAMSI), which is based in Duke University, in the Workshop on the Interface of Statistics and Optimization (WISO) in February 2017,
- Centre for Mathematical Studies and their Applications, which is based in École Normale Supérieure Paris-Saclay, in September 2017,
- Isaac Newton Institute for Mathematical Sciences, which is based in the University of Cambridge, in the workshop on Surrogate models for UQ in complex systems, in February 2018,
- SIAM Conference on Uncertainty Quantification, based in California, in April 2018.

The initial meetings and discussions about the work that has been done in Chapter 5 were funded by EPSRC as part of the "Mathematics for Living with Environmental

Change" call. Specifically, EPSRC funded the project *Research on Changes of Variability and Environmental Risk* (ReCoVER) aiming the use of novel mathematical tools for understanding, predicting and managing the effects of environmental change. I applied and secured, as a Principal Investigator, a mini-project grant from the EPSRC network ReCoVER. The work in Chapter 5 is done in collaboration with Dr Volker Roeber who is based in the University of Pau and Pays de l'Adour in France.

The work contained in the thesis has been collected in the following papers:

- Mathikolonis, T., Guillas, S. (2019), Surrogate-based Optimization using Mutual Information for Computer Experiments (optim-MICE). (*submitted* to the Journal of Global Optimization),
- Mathikolonis, T., Roeber, V., Guillas, S. (2019), Computationally efficient surrogate-based optimization of coastal storm waves heights and run-ups. (*submitted* to the Proceedings of the Royal Society A).

The developed optimization algorithm will be part of the Uncertainty Quantification (UQ) software and available on the Multi-Output Gaussian Process (MOGP) platform in GitHub. Both, the UQ software and the MOGP platform, are under active development by the Research Engineering Group at the Alan Turing Institute and aim to incorporate various activities related to uncertainty quantification in computer experiments.

Chapter 2

Background

2.1 Derivative - Free optimization

The standard mathematical approach in an optimization problem is to use all the available information contained in the derivatives of any function. However, this is not always the case as many practical applications require to optimize a function f over a domain of interest where derivatives are unavailable, unreliable or computationally prohibitive. For instance, f can be very expensive to compute or may have discontinuous derivatives. These problems are usually referred as *derivative-free optimization* or *black-box optimization* since the analytic form of the function is not known.

The development of different derivative-free algorithms starts in the mid-1960s when the Nelder-Mead (NM) simplex algorithm is invented. The NM algorithm is the most widely used direct search method for solving unconstrained optimization without derivatives (Nelder & Mead, 1965). Since then, a lot of studies have been done in the area with significant improvements by either providing convergence proofs (Lucidi & Sciandrone, 2002; Abramson & Audet, 2006; Conn et al., 2009b; Boukouvala et al., 2016) or using approximations of the expensive computer simulator, known as surrogate models, as well as incorporating the Bayesian optimization framework introduced by Kushner (1964) where the underlying function

is a realization of some stochastic process.

Derivative-free algorithms are in a great demand as they can cope with various challenges that are common in a lot of different fields such as expensive function evaluation, a black-box function, noise and uncertainty in computer simulation output and hidden constraints. Some of the areas where they have been applied are in physics (Zhao et al., 2006), energy security (Ciaurri et al., 2010), medical image registration (Oeuvray & Bierlaire, 2007) and environmental statistics (Stefanakis et al., 2014). Despite the fact that many derivative-free algorithms, such as the Genetic Algorithm and Particle Swarm optimization, have been proved to be reliable techniques for finding the global optimum, they often need a large number of function evaluations (Conn et al., 2009a), and therefore a lot of computational resources, which renders them unaffordable for computationally expensive problems.

Derivative-free algorithms can be classified into different categories (Rios & Sahinidis, 2013):

- *direct* vs. *model-based*: In direct methods, search directions (where to search for the optimal value) are determined using the values computed directly from the function f whereas in model-based methods, a surrogate model is built and utilized for the search process,
- *local* vs. *global*: Local optimization algorithms search only the nearby neighbourhood until a better - from the current - configuration is found, whereas global optimization methods explore the entire search space and exploit the search history until the optimal value is reached,
- *stochastic* vs. *deterministic*: Due to some randomness introduced, the output of a stochastic model is always different even though the same parameter values are used. On the other hand, a deterministic model, given particular parameter values, will always produce the same output.

Comprehensive reviews of the different derivative-free algorithms can be found in Rios & Sahinidis (2013), Kolda et al. (2003) and Conn et al. (2009a). The focus of the current work is on the *model-based methods* where a surrogate model takes the place of a $deterministic$ function for $global$ optimization purposes.

Bayesian Optimization is a derivative-free optimization framework for sequentially optimizing a black-box function. Although the Bayesian strategy was introduced over 50 years ago (Kushner, 1964), it has recently gained more attention in the machine learning community, especially after the development of the Efficient Global Optimization (EGO) algorithm proposed by Jones et al. (1998). In the current study, we do not fully adopt the Bayesian Optimization framework, but we follow its two-step approach. Firstly, we build a probabilistic model based on the assumptions made about the function being optimized and then, we choose an acquisition function, also known as loss function, which is used to determine at which input point the unknown function will be evaluated next. As in Bayesian Optimization, at each step of the optimization process, the probabilistic model is refined by considering the new data observed. Recent tutorials of Bayesian Optimization can be found in Shahriari et al. (2015) and Frazier (2018).

The most well known probabilistic model used in black-box optimization is *Gaussian Process* (GP) regression, which is also used as a *surrogate model* for computer experiments due to its flexibility and tractability (Snoek et al., 2012). In terms of the acquisition function, there is a huge range of strategies that can be used, however, we choose to focus on the idea of balancing the exploration - exploitation trade-off using the *Upper Confidence Bound* (UCB) algorithm. The sections below give a solid background for all the important components incorporated in the current research.

2.2 Surrogate modelling

Computer models, which are most of the time black-box functions, are widely used to study physical processes (Santner et al., 2003). To represent well a real-world system, computer models can often become very complex. This makes the optimization of a black-box function and further studies, such as sensitivity and uncertainty analyses, too costly and time-consuming since a large number of evaluations might be needed. To overcome this issue, a statistical surrogate model is built, also known as emulator or a meta-model, which replaces and accurately represents the computer model (Santner et al., 2003; O'Hagan, 2006).

Specifically, a statistical surrogate model, which was first introduced by Sacks et al. (1989) and Currin et al. (1991), is used as means for designing and analysing computer experiments (Santner et al., 2003). The idea is that we want to know the output of the computer model at different input values but, due to its computational cost, we build a surrogate model that approximates the input-output behaviour and accurately represents the analytical model. Through the years, surrogate models have been expanded into a standard statistical approach for predicting the output of a complex mathematical function with a lot of advancements (e.g. developing a surrogate model under a Bayesian framework (O'Hagan, 2006), emulating multivariate simulators (Conti & O'Hagan, 2010; Overstall & Woods, 2016), modelling the surrogate-model errors in the context of dynamical systems (Trehan et al., 2017)) and applications (e.g. epidemiology (Willem et al., 2014; Williams et al., 2019), environmental engineering (Sarri et al., 2012; Stefanakis et al., 2014), chemical engineering (Quirante et al., 2015; Bowman & Woods, 2016)).

Surrogate models can be classified into *physical* and *functional* (Conn et al., 2009a). Physical models are the surrogate models that are built from a physical or numerical simplification of the true functions, whereas functional are algebraic representations of the true functions. In the current work, we use functional surrogate models because previous knowledge of the physical system is not required, they are often

computationally inexpensive and are more suited for global optimization purposes as they might incorporate a stochastic component (Koziel et al., 2011; Conn et al., 2009a). There are different types of functional surrogate models that could be used such as polynomials interpolation (Giunta et al., 1997), multivariate adaptive regression splines (Friedman, 1991), radial basis functions (Gutmann, 2001) and Gaussian Process (GP) regression, well-known in geostatistics community as Kriging models (Currin et al., 1991).

In our work, we build and utilise a GP emulator. GP regression is the most popular statistical surrogate model as it allows modelling complicated functional forms. It not only offers a prediction at a new input point but also provides an estimate of the uncertainty in that prediction (Sacks et al., 1989). Rasmussen & Williams (2005) provides an extensive discussion of the GP and its properties. In the next section, we give a short description of the key components of a GP regression.

2.3 Gaussian Process (GP) regression

A GP is a continuous extension of multidimensional normal distribution. It is one of the most common stochastic processes used in derivative-free optimization, and computer experiments in general, as we can take advantage of all the convenient mathematical properties of a normal distribution. Under a Bayesian framework, a GP represents the prior knowledge about the unknown function f which is treated as a random function. In a non-Bayesian framework, f is treated as if it is drawn randomly from some population of functions, and one assumes that the distribution of functions in that population is a GP model.

Whereas a probability distribution describes random variables which are scalars or vectors and is used to model finite collection of real-valued variables, the GP enforces implicit properties of a function without relying on any parametric assumptions and can be seen as its distribution. Instead of having a mean vector and

a covariance matrix as in normal distribution, a GP is defined by its mean function $m(x)$ and covariance function $c(x,x')$. Let f be a function mapping an input $x \in \mathscr{X}$ into an output $y = f(x)$ in $\mathbb{R}$. An unknown function f is a GP if for any set of n inputs $\{x_1,\ldots,x_n\}$, the joint distribution of the set of outputs $\{f(x_1),\ldots,f(x_n)\}$ follows a multivariate normal distribution. Therefore, it can be thought as

$$f(x) \sim GP(m(x), c(x,x')), \tag{2.1}$$

$$\text{where } m(x) = \mathbb{E}[f(x)], \tag{2.2}$$

$$\text{and } c(x,x') = \mathbb{E}\left[(f(x) - m(x))\left(f(x') - m(x')\right)\right], \tag{2.3}$$

where the mean function $m(x)$ reflects the expected function value at input x and the covariance function $c(x,x')$ models the dependence between the function values at input points x and x'.

The mean function usually takes the form

$$m(x) = h(x)^T \beta, \tag{2.4}$$

a combination of a set of regression functions $h(x)$ with a vector of unknown coefficients β. It gives the expected value of the computer model output as a function of the inputs and shows an indicator of how the output broadly responds to changes in the inputs.

Two common forms used for the mean function are either to keep it constant for all input values, which indicates no prior knowledge about the computer models' output, or use a polynomial regression, which states that the output will be approximately linear in all the inputs. A quadratic form and possibly higher order polynomial terms can also be used if a non-linear trend is expected between the output of the computer model and each input variable (Bastos & O'Hagan, 2009). In practice, both forms tend to perform well (Rasmussen & Williams, 2005; Aye &

Heyns, 2018). As in most of the studies, so in the current one, the mean function is set to zero so as to let the variation in the data be explained by the covariance function.

The covariance function is defined as $c(x,x') = \sigma^2 K(x,x')$: a product of the process variance, which is a scalar parameter ($\sigma^2 > 0$), and a correlation function K. The process variance states the overall variance of the output at any input point (large values allow more variation). The correlation function states the strength of the relationship between the two configurations x and x'.

The covariance function - also known as kernel - is at the heart of the GP as it encodes the properties of the unknown function e.g. whether it is a relatively smooth and continuous function. Intuitively, the covariance function is used to define the similarity or the mutual informativeness of the function values $f(x)$ and $f(x')$ as a function of the input points x and x'. Unlike the mean function which can be chosen freely, an arbitrary covariance function, in general, is not valid as it has to be a symmetric positive semi-definite function.

In general, it is expected that input points x and x' which are sufficiently close in the input space $\mathscr{X}$ have similar $f(x)$ and $f(x')$ and higher correlation than input points that are far apart. Therefore, to define the similarity between two input points it is sufficient enough to measure their distance. Here, we will only refer to the most commonly-used covariance functions but a range of different class of covariance functions can be found in Rasmussen & Williams (2005).

Among the most popular covariance functions are the squared exponential, separable power exponential and Matérn functions. All these functions are characterised as *stationary* and/or *isotropic*. By definition, a covariance function is called *stationary* if it is invariant to translations of the inputs (e.g. it is a function of a distance $x-x'$ for any pair of inputs x and x') and *isotropic*, if it is a function of the Euclidean

distance $\|x-x'\|$ (not the direction). We define below the covariance functions:

- *Squared exponential* is probably the most popular covariance function within the machine learning field. The functions sampled from a GP are infinitely differentiable which means that the GP with this covariance structure has mean square derivative of all orders and therefore, is very smooth (Rasmussen & Williams, 2005). The squared exponential covariance function is defined as

$$K(x,x'|l) = \exp\left\{-\frac{\|x-x'\|^2}{l}\right\}, \tag{2.5}$$

  with parameter l defined as the characteristic length-scale: the range at which x and x' are close enough to influence each other significantly. In practice, length-scale value describes how smooth a function is. Small values indicate that the function values can change quickly and vice-versa.

- *Separable power exponential* is a standard choice of modelling computer experiments especially in higher dimensions (Santner et al., 2003; Gramacy & Lian, 2012). This covariance function is useful when dealing with massive data sets as it reduces the computational time (Genton, 2001).

  Its structure is built based on the property of multiplication of covariance functions, which yields also a covariance function, by taking the product of correlations across each dimension $i = 1,\ldots,d$. Even if the core part of the covariance function is based on the squared exponential function, the resulting function is still stationary but not isotropic because the characteristic length-scale is not the same for all the inputs. Its advantage can be fully seen once a nugget parameter is added in the calculations. A nugget parameter (a scalar positive value) is mainly added in the correlation function with the aim to introduce a measurement error into a stochastic processes and/or to prevent the $\mathbf{K}$ from becoming numerical singular. A well-conditioned correlation matrix [1] can then be achieved and this will lead us to a better predictive accuracy

[1] A well-conditioned matrix is defined as a matrix whose inverse can be computed with good accuracy.

(Gramacy & Lian, 2012; Bilionis et al., 2013). The form of the separable power exponential covariance function is

$$K(x,x'|\xi)) = \prod_{i=1}^{d} \exp\left\{-\frac{\|x-x'\|^p}{l_i}\right\}, \tag{2.6}$$

where $\xi = (l_1,\ldots,l_d)^T \in \mathbb{R}_+^d$. Here, l_i is defined as the characteristic length-scale for the ith input dimension. The degrees of smoothness is controlled by $0 < p \leq 2$ with a typical default choice 2 as the function is infinitely differentiable (Gramacy, 2007).

- *Matérn* is also one of the most popular covariance functions. Stein (1999) states that the strong smoothness assumption in the squared exponential covariance function is unrealistic and suggests that a covariance from the Matérn family is a better choice for spatial data and computer experiments. The general form of the Matérn covariance function is

$$K^*(x,x'|\xi,\nu) = \prod_{i=1}^{d} \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{2\nu^{1/2}\|x-x'\|}{l_i}\right)^{\nu} J_\nu \left(\frac{2\nu^{1/2}\|x-x'\|}{l_i}\right), \tag{2.7}$$

where $\xi = (l_1,\ldots,l_d)^T$ is the characteristic length-scale vector which measures how quickly the correlations decay with distance ($l_i > 0$), ν is a positive parameter which specifies the degree of smoothness, Γ_ν is the Gamma function for ν and J_ν is a modified Bessel function of order $\nu > 0$.

The great flexibility of the Matérn covariance function compared to the squared exponential covariance function is the parameter ν which controls the degree of smoothness (Handcock & Stein, 1993; Minasny & McBratney, 2005; Rougier et al., 2009). Generally, the larger the ν, the smoother the process. Two of the most commonly-used cases in GP are when $\nu = 3/2$ and $\nu = 5/2$ as the smoothness is kept at a normal level. For $\nu \to \infty$ the covariance function converges to a squared exponential covariance function.

2.3.1 Determination of hyper-parameters

To be able to use the GP as an emulator for an expensive computer model and replicate the functional connection between inputs and outputs, it is necessary to estimate all the parameters from the data collected by running the computer model. The parameters considered in a GP regression are the regression coefficient β, the process variance σ^2 and the characteristic length-scale (also known as the correlation length) l or ξ, if for each ith input dimension a different correlation length is considered. These parameters are referred to as *hyper-parameters* and can be estimated using different statistical modelling approaches e.g. Maximum Likelihood Estimate (MLE) method, Restricted Maximum Likelihood method and Bayesian methods. A short description of the two most commonly used approaches are provided below.

Bayesian Inference

A solid foundation for Bayesian Inference can be found in Lee (2004) and Gelman et al. (2013). Here, we only give the basic idea of its framework. The hyper-parameters, defined as $\theta = (\beta, \sigma^2, l)^T$, are treated as unknown and random variables. Suppose that $y = (y_1 = f(x_1), \ldots, y_n = f(x_n))$ contains n realizations of the computer model at the input points $x_1, \ldots, x_n$. Before seeing the data y, any prior knowledge we have about the hyper-parameters is specified using a prior distribution $p(\theta)$ over the hyper-parameters. The prior distribution is modified once the data is observed to produce the posterior distribution $p(\theta|y)$. The posterior distribution is calculated based on the Bayes' theorem and combines the information from prior knowledge and data (through the likelihood function $p(y|\theta)$):

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}. \tag{2.8}$$

For algebraic convenience, we can specify our prior knowledge using a *conjugate prior*, a prior which when combined with the likelihood function, produces a posterior distribution that belongs to the same family as the prior distribution. Conjugate priors are particularly convenient because they lead to a closed-form expression for the posterior. However, in certain cases, the form of the posterior distribution can-

not be found and as a result, in order to evaluate the posterior distribution and make predictions, it is necessary to calculate the marginal likelihood $p(Y)$, known as the model evidence:

$$p(Y) = \int p(y|\theta)p(\theta)\, d\theta. \tag{2.9}$$

Computing analytically the integral for high-dimensional posterior distributions is often intractable and as a result, Markov Chain Monte Carlo (MCMC) methods are used (details about the MCMC method can be found in Gilks et al. (1995) and Andrieu et al. (2003)). The MCMC method is the standard computational method to compute a high-dimensional posterior distribution through sampling and make inference about the parameters. Even if the MCMC algorithm is simple in structure and generally easy to implement, it can be computationally expensive and most Bayesian analysis can take longer. This is due to the convergence problem in which we need to decide when is the appropriate time to stop sampling and use the samples to estimate characteristics of the distribution of interest.

Using the Bayesian method, we take into account the uncertainty related to parameter values on subsequent predictions and combine our prior knowledge with data. Due to that, a Bayesian approach becomes very attractive when calculating the GP hyper-parameters, sometimes under fully Bayesian setting (Handcock & Stein, 1993; Williams & Rasmussen, 1996; Williams & Barber, 1998; Gramacy & Lee, 2009) and otherwise using empirical Bayes methods (Oakley, 1999; Rasmussen & Williams, 2005; Garbuno-Inigo et al., 2016).

Maximum Likelihood

The Maximum Likelihood Estimation (MLE) method is also widely used to estimate the hyper-parameters of a GP. In this section, we give a brief description of the design and analysis of computer experiments (DACE) framework, proposed Sacks et al. (1989). This framework is also used in the current study.

The MLE method aims to find an estimator for each GP hyper-parameter that

is most consistent with the observed data. Assuming that the distribution of the computer model output $y = \{y_1 = f(x_1), \ldots, y_n = f(x_n)\}$, which resulted from a set of possible input points $X = \{x_1, \ldots, x_n\}$, follows the multivariate normal distribution. We choose the values of hyper-parameters that maximize the likelihood function given n observations of y. For computational convenience, we take the multivariate normal log-likelihood function:

$$\ell(\theta|y) = -\frac{n}{2} ln(\sigma^2) - \frac{1}{2} ln(|K|) - \frac{(y - H\beta)^T K^{-1} (y - H\beta)}{2\sigma^2}, \tag{2.10}$$

where $|K|$ denotes the determinant of K. The MLE of β, which is a d x 1 vector of unknown regression coefficients, is its generalized least squares estimate

$$\hat{\beta} = (H^T K^{-1} H^T)^{-1} H^T K^{-1} y, \tag{2.11}$$

where H is the n x d matrix of regression functions and K is the n x n correlation matrix. The MLE solution for the process variance σ^2 can be expressed as:

$$\hat{\sigma}^2 = \hat{\sigma}^2(\xi) = \frac{1}{n} (y - H\hat{\beta})^T K_\xi^{-1} (y - H\hat{\beta}), \tag{2.12}$$

where $\xi = (l_1, \ldots, l_d)$ is a vector that represents the correlation length for the ith input dimension and is set as fixed (Sacks et al., 1989). The subscript ξ in the term K_ξ is included to emphasize the fact that the correlation matrix is a function of the correlation length parameter. A closed-form solution for ξ does not exist and a numerical optimization is required. To reduce the computational burden needed for the optimization, we substitute the known optimal values of $\hat{\beta}$ (2.11) and $\hat{\sigma}^2$ (2.12) into the log-likelihood function (1.10). The MLE of ξ, denoted by $\hat{\xi}$, can be found by maximizing the profile log-likelihood function for ξ (Mardia & Watkins, 1989),

$$\ell(\hat{\beta}, \hat{\sigma}^2 | \hat{\xi}) = -\frac{n}{2} \ln(\hat{\sigma}^2(\xi)) - \frac{1}{2} \ln(|K_\xi|) - \frac{n}{2}, \tag{2.13}$$

which means finding the values of ξ that minimize (Santner et al., 2003),

$$n \ln \hat{\sigma}^2(\xi) + \ln(|K_\xi|). \tag{2.14}$$

The MLE optimization can be computationally expensive and in some cases, we might need to address the issue of having multiple local maxima. To avoid that issue, the current work uses the genetic algorithm approach which is robust despite the fact that it can be computationally intensive for likelihood optimization. However, what makes the MLE method so popular is its optimal asymptotic properties. There is a fair amount of literature on the estimation of hyper-parameters of a GP using the MLE method (Forrester et al., 2008; Ginsbourger et al., 2009; Bachoc, 2013; Beck & Guillas, 2016). A classical derivation of the MLE approach and its properties can be found in Mardia & Marshall (1984), Sacks et al. (1989) and Stein (1999).

2.3.2 Prediction with Gaussian Process

In computer experiments, the computer model output $y = f(x)$ is assumed to be a deterministic real-valued function of the d-dimensional variable $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$. The unknown function $f(x)$ is treated as a random function only at the input points where the computer model has not been evaluated yet. Specifically, the unknown function is modelled as a random field given the training data-set; a set of input-output pairs included in the training data-set at time T.

Assuming that the random function is a Gaussian Process is a great advantage as a GP regression not only offers a prediction at a new input point but also provides an estimate of the uncertainty in that prediction (Sacks et al., 1989). For predicting the output y of the computer model at any input value, the GP emulator is used. Conditionally on the training outputs after T iterations and on the estimated hyper-parameters, $Y_T = [y_t, \ldots, y_T]^T$ at points $X_T = \{x_t, \ldots, x_T\}$, the process is still a GP and the predictive distribution of the output at a new input points x, also known as test points, is a multivariate normal with mean $\hat{y}(x)$ and variance $\hat{s}^2(x)$:

$$\hat{y}_{T+1}(x) = k_T(x)^T (K_T + \sigma^2 I)^{-1} Y_T \tag{2.15}$$

$$\text{and } \hat{s}^2_{T+1}(x) = k(x,x) - k_T(x)^T (K_T + \sigma^2 I)^{-1} k_T(x) \tag{2.16}$$

where $k_T(x) = [k(x_1,x),\ldots,k(x_T,x)]^T$ is the vector of covariances between the input points already chosen and x and $K_T = [(x,x')]_{x,x' \in X_T}$ is the covariance matrix.

The Eq. 2.15 and 2.16 are derived by following the approach proposed by Sacks et al. (1989). In general, to predict the output of the computer model, linear predictors of the form $\hat{y}(x) = \lambda^T(x)y$ for some $\lambda(x) \in \mathbb{R}^n$ are used. Assuming that the hyper-parameters are known, the mean $\hat{y}(x)$ is the best linear unbiased predictor (BLUP) which minimizes the mean square error for the prediction (MSPE) with respect to $\lambda(x)$. The uncertainty in the prediction, or otherwise the variance of the predictive distribution, is given as $\hat{s}(x) = \text{MSE}[\hat{y}(x)]$.

2.4 Design of Experiments

The computer experiments involve running a deterministic black-box function where its output is not affected by any uncontrollable variable and thus, there is no random error. Therefore, techniques such as replication and blocking, which are used to estimate and control the magnitude of random error, are not needed. Nevertheless, a considerable level of uncertainty arises due to the lack of knowledge about the relationship between inputs and outputs. The input-output behaviour is approximated through the statistical surrogate model. Since the computer model is computationally expensive to run and sometimes only a limited number of runs is affordable, it is important to choose an experimental design that minimizes the computational cost, but maximizes the information about the computer model.

Experimental design is defined as the process of selecting an efficient set of input points (n design points) at which to evaluate the computer model and compute

the output (Santner et al., 2003). The design points are combinations of the input variables and are used as the training input points to predict the output. They are chosen strategically based on various optimization criteria e.g. mean squared prediction error (MSPE) (Sacks et al., 1989), maximum entropy sampling (Currin et al., 1988) and distance between design points (Johnson et al., 1990). There are different experimental design strategies in the literature that are mainly classified into two categories: space-filling designs and adaptive designs.

2.4.1 Space-filling designs

The design points are spread throughout the input space, since all regions are treated as equally important, giving the opportunity to obtain an overview of what the response surface might look like. They are all chosen before computing any function evaluation without concentrating in clusters or at the corners of the input space. The main issue is that when certain regions of the input space are not important, a certain amount of computational time is wasted.

Examples of space-filling designs are uniform designs, multilayer designs, maximin and minimax distance designs and Latin hypercube designs (LHD) (Sacks et al., 1989; Simpson et al., 2001; Pronzato & Müller, 2012). In terms of the prediction accuracy, all the designs perform well (Marin, 2005) however, we will only give a short description about LHD, which is commonly used in the literature but also in the current study.

- Latin hypercube design has been proposed by McKay et al. (1979) as an alternative method for choosing simulator inputs to regular grid design, with the aim to avoid the grid's collapsing property: no two LH design points share the same value for any parameter.

  As in grid design, which is the simplest possible experimental design, in LH each input range is partitioned into n equal probability and non-overlapping intervals. Unlike a grid, the number of partitions is equal to the number of

the design points selected. Then, one value from each interval is randomly selected with respect to the probability density of the interval. The same process is followed for each dimension d, for all the inputs $x_1, \ldots, x_d$. The n values selected for x_1 are paired randomly with the n values for x_2 until the nd - tuplets are formed. Fig. 2.1 shows an example of LHD with two parameters.

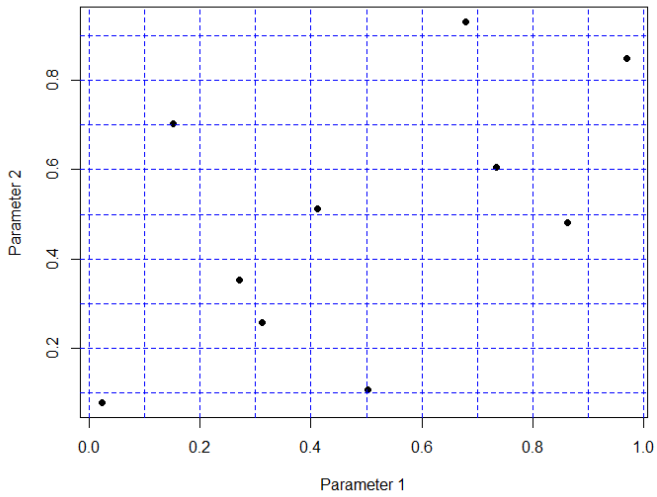


Figure 2.1: An LHD with two dimensions and 10 points

Unlike grids, the number of partitions in LHD does not need to grow exponentially with the dimensionality of the input space (Urban & Fricker, 2010). Since the design points are chosen before learning about the functional input-output behaviour, it is important to ensure that the design should be space-filling so as to collect information from the entire input space. Maximin- and minimax- distance criteria can be used as an additional constraint while an LHD is constructed. In a maximin-LHD the points are spread over the input space so that the minimum distance among the design points is maximised, whereas in minimax-LHD, the maximal distance from any point is minimized (Johnson et al., 1990). In practice, the maximin distance criterion pushes the points toward to the design boundaries while the minimax distance criterion tends to push more points in the interior of the input space (Ba & Joseph,

2011). The maximin design are less computationally intensive than minimax designs.

2.4.2 Adaptive designs

To address the issue of having design points over some unnecessary regions of the input space, adaptive designs have been proposed as efficient alternatives to space-filling designs (Santner et al., 2003; Lam, 2008; Gramacy & Lee, 2009; Beck & Guillas, 2016). The advantage of adaptive designs is that they collect information about the unknown function during the experimental design process in the form of input-output data from simulation runs. The design points are chosen sequentially, often one-at-a-time or by batches, from regions where uncertainty is large. Compared to space-filling designs, the adaptive designs can be computationally expensive and sometimes time-consuming however, they can often be effective in practice (Beck & Guillas, 2016). Examples of adaptive designs, which are going to be further discussed below, are active learning MacKay (ALM), active learning Cohn (ALC) and Mutual Information for Computer Experiments (MICE).

To best approximate the computer model output over the design space $\mathscr{X} \subseteq \mathbb{R}^d$, it is important to determine the input values at which the data should be collected. To end up with an optimal design, a specific design criterion needs to be optimized at each step of the process. Since the design points are chosen sequentially, only the most informative points are included in the training data set. A common characteristic of the adaptive designs described below is that all are built to measure an information gain quantity. Some of them are based on a simple measure of the predictive uncertainty, whereas others are based on the classical information theoretic measures such as entropy and mutual information.

For convenience, X_G is a discrete design space, (G for Grid, $X_G \subseteq \mathscr{X}$) with n_G number of points, initial design is defined as (X_k, y_k) with k number of points, X_{cand} ($X_{cand} \subseteq X_G$) is defined as the set of n_{cand} candidate points.

- *ALM*: At stage k the algorithm, proposed by MacKay (1992), chooses the next design point x_{k+1} from X_{cand} that maximizes the predictive variance of the GP,

$$x_{k+1} = \underset{x \in X_{cand}}{\arg\max}\, \hat{s}^2(x). \tag{2.17}$$

  The computational demand of ALM depends on how large the set of candidate points is. Studies show that ALM is not as computational expensive as other sequential designs, and because it is easy to implement is often preferred over others approaches (Gramacy & Lee, 2009; Bilionis et al., 2013; Beck & Guillas, 2016). However, ALM tends to place many points on the boundaries of the design space, especially in high-dimensional space. As the dimension size d is increased, the number of boundary points grows as well. Boundary points are optimal only when the model is known precisely. Since the full form of the model is not known in advance, having points at the boundaries might be suboptimal (Chaloner & Verdinelli, 1995; Gramacy & Lee, 2009). Various studies also state that the boundary points are less informative than nearby interior points (Chaloner & Verdinelli, 1995; Krause et al., 2008; Gramacy & Lee, 2009; Beck & Guillas, 2016).

- *ALC*: The algorithm, proposed by Cohn (1996), considers the effect of each candidate point on the entire input space and the existing information gained up to the current stage. As in ALM, it is based on the predictive variance of the candidate points. ALC sequentially selects the next design point x_{k+1} that yields the largest reduction in predictive variance over the input space. This is defined as,

$$x_{k+1} = \underset{x \in X_{cand}}{\arg\max} \int_{\mathcal{X}} (\hat{s}^2(x') - \hat{s}^2_{k \cup x}(x'))\, dx', \tag{2.18}$$

  where $\hat{s}^2(x')$ is the variance of the design point x', which is already in the training data set, before observing the output at x_{k+1} and $\hat{s}^2_{k \cup x}(x')$ is the variance at x' when the new point x_{k+1} is added in the design. In practice, the integral is often approximated by a sum over a reference set, a grid of n_{ref}

reference points in the input space, that is,

$$x_{k+1} = \underset{x \in X_{cand}}{\arg\max} \frac{1}{n_{ref}} \sum_{i=1}^{n_{ref}} (\hat{s}^2(x_i) - \hat{s}^2_{k \cup x}(x_i)). \tag{2.19}$$

Compared with the ALM, ALC is computationally more expensive. It performs better as it examines the effect of each point from the candidate set on the entire domain and has less concerns about the uncertainty near the boundaries of the input space (Seo et al., 2000; Gramacy & Lee, 2009).

- *MICE*: The algorithm, proposed by Beck & Guillas (2016), is a modified version of the mutual information (MI) criterion involved in the sequential algorithm given by Krause et al. (2008). As with the MI criterion, MICE is based on the idea of entropy and mutual information, two key measures of information theory.

  The required definitions are only briefly given but a good overview of the information theory can be found in Cover & Thomas (2006). Entropy is defined as the uncertainty of a single random variable or otherwise, it measures the amount of information required to describe the random variable. Mutual information is described as the reduction in the uncertainty of one random vector due to the knowledge of the other. In other words, it measures the amount of information that one random vector contains about another random vector. To express the mutual information of two vectors, we use the relative entropy, also known as Kullback-Leibler distance, which is a measure of the distance between the distributions of the two vectors.

  Suppose that the two random vectors Y and Y' have a joint probability density function $p_{Y,Y'}(y,y')$ and marginal probability density functions $p_Y(\mathbf{y})$ and $p_{Y'}(\mathbf{y}')$. As given in Cover & Thomas (2006), the mutual information of the

two vectors, denoted as $I(Y;Y')$, is defined as

$$I(Y;Y') = \int p_{Y,Y'}(y,y') \log \left(\frac{p_{Y,Y'}(y,y')}{p_Y(y)p_{Y'}(y')} \right) dydy'. \tag{2.20}$$

The definition of mutual information can be rewritten in such a way that shows the relationship between entropy and mutual information of the two random vectors. Therefore, mutual information is defined as

$$I(Y;Y') = H(Y) - H(Y|Y'), \tag{2.21}$$

where $H(Y)$ is the entropy, also known as self-information [2], of the random vector Y, and $H(Y|Y')$ is the conditional entropy of Y given Y'.

The idea of using the information theoretic mutual information measure in the context of experimental designs and in spatial statistics is first used by Caselton & Zidek (1984) with the aim to overcome the 'waste information' property of the entropy criterion as sensors are placed along the boundary of the input space. In contrast to the entropy criterion, where only the uncertainty of the selected sensor locations is considered, the MI criterion searches for the subset of sensor locations that most significantly reduces the uncertainty about the locations in the rest of the space.

The objective is to find an optimal monitoring network design X_n^* with n input points that maximizes the MI between the two sets: the set of points already in the design X_n and the set with the rest points left in the discrete design space $X_G \setminus X_n^*$,

$$X_n^* = \underset{X_n \in X_{cand}}{\arg\max}\, I\left(X_G \setminus X_n; X_n\right) \tag{2.22}$$

$$= \underset{X_n \in X_{cand}}{\arg\max}\, H\left(X_G \setminus X_n\right) - H\left(X_G \setminus X_n | X_n\right). \tag{2.23}$$

[2] The mutual information of a random variable with itself is the entropy of the random variable.

Specifically, we want to find the set X_n^* that maximally reduces the entropy over the rest of the space $X_G \setminus X_n^*$ (Krause et al., 2008). Maximizing the MI between the two sets is defined as an NP-hard problem[3].

In order to avoid solving directly the optimization problem (Eq. 2.22), Krause et al. (2008) proposed a greedy algorithm where an optimal design can be achieved by adding sensors in sequence, one-at-a-time. More formally, at each stage k, the new input point $x \in X_{cand}$ added in the design X_k is the one that maximizes the difference:

$$\underset{x \in X_{cand}}{\arg\max}\ I\big((X_k \cup x); X_G \setminus (X_k \cup x)\big) - I\big(X_k; X_G \setminus X_k\big). \tag{2.24}$$

For GPs, the optimisation problem can be written as

$$x_{k+1} = \underset{x \in X_{cand}}{\arg\max}\ \hat{s}_k^2(x) / \hat{s}_{G\setminus(k\cup x)}^2(x), \tag{2.25}$$

where $G \setminus (k \cup x)$ denotes the $X_G \setminus (X_k \cup x)$, the set with the input points that have not been selected yet.

The efficiency and prediction accuracy of the MI criterion is demonstrated in Krause et al. (2008) where the greedy algorithm is compared with the classical experimental design criteria. However, the quality and robustness of the MI design is strongly affected by the set of candidate points and how they appear in the input space. For example, when the points are irregularly spaced and/or in clusters or in a non-equidistant grid the MI criterion is not robust (Beck & Guillas, 2016). In MICE, the robustness is improved by adding in the sequential algorithm an extra parameter. Specifically, the MICE criterion is defined as

$$x_{k+1} = \underset{x \in X_{cand}}{\arg\max}\ \hat{s}_k^2(x) / \hat{s}_{G\setminus(k\cup x)}^2(x; \tau_s^2), \tag{2.26}$$

[3] An optimization problem is defined as NP-hard if an exact solution requires computational time which grows exponentially with the size of the problem.

where τ_s^2, referred to as the nugget parameter, is the extra parameter added in the correlation matrix K of the GP on $X_G \setminus (X_k \cup x)$. Using MICE to construct a sequential design of a computer experiment, predictions become smoother and the GP's variance is flatter (Beck & Guillas, 2016). This is mainly due to the addition of the nugget parameter which, in general, can improve numerical stability when the computer experiments are deterministic (Gramacy & Lee, 2009). In theory, the nugget parameter can take any positive value $\tau_s^2 > 0$. In practice, an ideal value is close to 1 as shown explicitly in Beck & Guillas (2016).

2.5 Exploration and exploitation in GP framework

The aim of the current work is to find the maximum of an unknown function $f : \mathscr{X} \to \mathbb{R}$, where $\mathscr{X} \subseteq \mathbb{R}^d$, with the lowest possible number of function evaluations. This is denoted as:

$$x^* = \arg\max_{x \in \mathscr{X}} f(x). \tag{2.27}$$

At each iteration t, we choose the next input point x_t for evaluation based on the knowledge obtained through the GP emulator built from the previous function evaluations. Once the unknown function is evaluated at the chosen point, the new point is added in the design and the training data is updated. Then, the GP model, which captures the characteristics of the unknown function, is refined using the new observed data. By repeating these steps, the true optimum of the unknown function can be achieved but not necessarily in the lowest possible function evaluations. Since each evaluation of the computer model is expensive, it is important to ensure that the next input point x_t is the maximum or will lead us to the maximum of the objective function within few iterations.

The ideal strategy to find the optimum value is to work with *exploration* and *exploitation*. Its basic idea is to gather more, or enough, information about the objective function by *exploring* the uncertain regions and then, make the best deci-

sion by *exploiting* (here optimizing) all the available information already known. It is generally accepted that the more uncertain we are about the function in a given region, the more information we can gather about the unknown function. Therefore, the next point x_t is chosen from regions with high mean and variance. However, to determine where to evaluate next and find the optimum, a balance between exploration and exploitation is required.

The trade-off between exploration and exploitation has been studied in various studies within machine learning (Črepinšek et al., 2013; Kaelbling et al., 1996; Ishii et al., 2002) and often seen as a multi-armed bandit problem (Auer et al., 2002; Bubeck et al., 2011; Robbins, 1985; Srinivas et al., 2010). Searching for a balance between exploration and exploitation is important as it affects the overall optimization performance such as its accuracy (Chen et al., 2009). The balance can be achieved using an *acquisition function*, a function that also help us to find the next point for evaluation. Both the acquisition function and the bandit setting are discussed in more details in the next pages.

2.5.1 Acquisition function

In theory, an acquisition function is used not only to control the exploration-exploitation trade off but also to guide us to search for the optimum (Brochu et al., 2010). It incorporates the estimates obtained from the statistical surrogate model about the computer model and the uncertainty at any given point. It is defined in such a way that obtaining high values of the acquisition function correspond to potentially high values of the objective function (Hoffman et al., 2011). This would happen because the prediction might be high and/or the uncertainty in that region is high.

The acquisition function depends on the already observed points and the GP emulator's output. Specifically, it is built based on the current best value $f(x^*)$, the predictive mean $\hat{y}(x)$ (2.15) and the predictive variance $\hat{s}(x)$ (2.16). A trade-off

parameter β is also added in the formulation of acquisition function in order to balance the exploration-exploitation trade-off. Here only a short description of the three most popular choices is given, but more details on the different acquisition functions available in the literature can be found in the comprehensive review of the Bayesian Optimisation given in Brochu et al. (2010) and Wilson et al. (2018).

- *Probability of Improvement* (PI) is built based on the early work of Kushner (1964). It evaluates the unknown function at the point where the improvement might be occur. Specifically, the point with the highest PI over the best point seen so far is selected. Under the GP, PI can be computed as:

$$\begin{aligned} \mathrm{PI}(x) &= P(f(x) \geq f(x^*) + \beta) \\ &= \Phi\left(\frac{\hat{y}(x) - f(x^*) - \beta}{\hat{\sigma}(x)}\right), \end{aligned} \tag{2.28}$$

  where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

  Without including the trade-off parameter β, only pure exploitation can be achieved and the chosen point will be the one that is greater than the current observed optimum and not necessarily the point with the greater impact and uncertainty (Brochu et al., 2010). On the other hand, if we set the trade-off parameter too high, the exploration part becomes stronger and as a result, the algorithm will be slow to fine-tune the optimal solution (Jones, 2001). The value of the trade-off parameter has been examined in various studies (Jones, 2001; Lizotte, 2008) however, the exact choice of β is always left to the user.

- *Expected Improvement* (EI), which is introduced by Mockus et al. (1978), is similar to PI. However, it takes into account not only the probability of improvement but also the magnitude of the improvement a point can potentially yield. Specifically, the point that maximizes EI, is the point that improves the unknown function the most. The EI can be evaluated analytically and its

closed form under the GP is:

$$\mathrm{EI}(x) = (\hat{y}(x) - f(x^*) - \beta)\,\Phi\left(\frac{\hat{y}(x) - f(x^*) - \beta}{\hat{\sigma}(x)}\right) + \hat{\sigma}(x)\,\phi\left(\frac{\hat{y}(x) - f(\mathbf{x}^*) - \beta}{\hat{\sigma}(x)}\right), \tag{2.29}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the probability density function and cumulative distribution function of the standard normal distribution respectively (Jones et al., 1998; Huang et al., 2006). As in PI, the value of the trade-off parameter β ($\beta \geq 0$) is left to the user, however Lizotte (2008) suggests that by setting $\beta = 0.01$ we can achieve a good balance in most of the cases.

- *Upper Confidence Bound* (UCB) is mainly used in Cox & John (1992) which suggests a sequential optimisation algorithm where the evaluation points are selected based on the confidence bounds. Specifically, the next point for evaluation is the one that maximizes the UCB. An established connection between GP optimization and UCB is given in Srinivas et al. (2010) and can be defined as:

$$\text{GP-UCB}(x) = \underset{x \in \mathscr{X}}{\arg\max}\ \hat{y}(x) + \beta^{1/2}\hat{\sigma}(x), \tag{2.30}$$

  Similar with PI and EI, the value of β is left to the user. Choosing a high β, the algorithm will focus its search on regions with high uncertainty.

Among the three acquisition functions, EI and GP-UCB are better-behaved than PI as the true global optimum of an unknown function can be found efficiently (Snoek et al., 2012). Different empirical studies show that either GP-UCB performs at least on par with EI or slightly better (Srinivas et al., 2010; Kandasamy et al., 2015; Wilson et al., 2018). The convergence rates have been extensively studied for both acquisition functions, EI and GP-UCB, but the theoretical properties of EI are still remained elusive (Vazquez & Bect, 2007; Dani et al., 2008a; Srinivas et al., 2010; Bull, 2011; Qin et al., 2017).

The current work uses the GP-UCB as an acquisition function. The UCB, and the confidence bounds in general, is proved that it can be successfully applied in complicated situations as well as deal with the exploration and exploitation in a bandit setting (Auer, 2002). More formally, the estimated mean of the computer model output guide us for further exploitation whereas the width of the confidence bounds, which reflects the uncertainty of the algorithm's knowledge at each time step, controls the exploration (Auer et al., 2002; Auer, 2002).

2.5.2 GP optimization and the bandit setting

A multi-armed bandit problem is a sequential decision making problem where at each time step of a time horizon T, the algorithm chooses one of the available arms (i.e. candidate points) and calculates its reward (Robbins, 1985). Depending on the chosen arm, it is assumed that the reward is sampled, independently from the previous rewards, from some fixed but unknown distribution. The aim is to maximize the total reward by optimally balancing exploration and exploitation: exploring all the available arms to get more information while exploiting the arm with the highest reward.

When GP optimization is formulated as a bandit problem, the value of the unknown function at the chosen point x_t is seen as the reward and the aim is to maximize the sum of rewards $\sum_{t=1}^{T} f(x_t)$. This follows the same scope as the equation (2.27) since at each time step t we choose to add in the design the input point x_t that gives the maximum reward. By sequentially optimizing the unknown reward function at each time step t, after T iterations the sum of rewards is also maximised. A standard performance metric of the whole strategy of a bandit problem is the *cumulative regret*, which measures the loss in reward due to not knowing the maximum value of f, and the objective is to minimize it after T iterations (Bubeck et al., 2012). To ensure that the strategy is performed well at each time step, the *simple regret* r_t is

calculated at each iteration t as:

$$r_t = f(x^*) - f(x_t). \tag{2.31}$$

Theoretical analysis of algorithms aiming to maximize the sum of rewards, under the multi-armed bandit setting, can be found in various studies (Azimi et al., 2010; Contal et al., 2013; Desautels et al., 2014; Grünewälder et al., 2010; Lai & Robbins, 1985; Srinivas et al., 2010).

2.5.3 Sequential algorithms: towards the parallel strategy

The most successful kriging-based optimization technique is the Efficient Global Optimization (EGO) algorithm proposed in Jones et al. (1998). At the beginning, a GP model is fitted based on an initial design set. The algorithm follows an iterative procedure where, at each iteration, a new candidate point is chosen by maximizing Expected Improvement (EI), known as a sampling criterion. Then, the GP is re-fitted again considering the updated design set. The traditional EGO algorithm is also extended into a parallel optimization scheme, by which multiple candidate points are chosen at each iteration. Such a strategy is the q-EI, proposed in Ginsbourger et al. (2008), where a batch of q points is added to the design set, at the same time, by maximizing an approximate expression of EI. The computational cost of the proposed algorithm is kept at the minimum as the kriging hyper-parameters are not re-optimized each time a point is added in the design.

The "adding several points at once" strategy gets great attention as a lot of different variants of the parallel EGO algorithm are proposed in the literature. Instead of using directly the Kriging method, Monte Carlo simulation is used for the multi-point EI estimation which is proved that is a reliable technique but with considerable extra computational cost (Janusevskis et al., 2012). A stochastic gradient algorithm is proposed in Wu & Frazier (2016) as an alternative approach of maximizing the q-EI, whereas in Zhan et al. (2017), the EGO-PEI algorithm, which is based on a

pseudo EI criterion, is suggested as an extension of the traditional EGO algorithm for choosing multiple design points for parallel evaluations.

A theoretical analysis of algorithms aiming to maximize the sum of rewards, or minimize cumulative regret, under the bandit setting on a certain class of UCB, has been initially studied by Lai & Robbins (1985). Specifically, asymptotically efficient allocation rules are constructed showing that the regret in a bandit problem has to grow logarithmically in the number of iterations. Since then, the asymptotic behaviour of various algorithms, which incorporate the UCB policy, has been studied in a number of studies but none of them were applicable in a GP framework (Lai & Robbins, 1985; Auer et al., 2002; Dani et al., 2008b; Kleinberg et al., 2008).

In contrast to EGO-based algorithms, where the convergence rates are under ongoing study (Bull, 2011; Vazquez & Bect, 2010; Qin et al., 2017), Srinivas et al. (2010) give the first theoretical bounds of cumulative regret, for functions sampled from a GP, which can be translated into convergence rates for GP optimization. A technical connection between the multi-armed bandit setting and the experimental design is also achieved as the regret is bounded by an information gain quantity used as a sampling criterion (Krause & Guestrin, 2012).

The new GP optimization algorithm chooses the new candidate points that maximize the UCB (Srinivas et al., 2010). At each iteration t, the point x_t is chosen from locations where the unknown function is uncertain (large predictive variance) and the maximum reward can be achieved (large predictive mean). The optimization of the unknown function f is formalized as a multi-armed bandit problem where the GP predictive uncertainty is used to control the exploration and exploitation. The performance of the GP-UCB algorithm is measured according to the regret, the difference between the actual maximum and the best result achieved, and the cumulative regret. The objective is to minimize the cumulative regret, therefore maximizing the black-box function.

The trade-off between exploration and exploitation for the contextual GP bandit problems is also addressed in the Contextual Gaussian Process Bandit Optimization (CGP-UCB) algorithm proposed by Krause & Ong (2011). The pay-off function corresponding to context-action pairs is modelled as a sample from a GP over the context-action space. In the context free setting (without considering the exploration/exploitation trade-off), Grünewälder et al. (2010) gives sharp bounds assuming that the entire horizon T is known and always the optimal arm is chosen rather than giving bounds on minimizing cumulative regret.

A multi-fidelity version of the GP bandit problem was investigated by Kandasamy et al. (2016) where the MF-GP-UCB algorithm, an extension of GP-UCB, aims to eliminate the low function value regions using cheap lower fidelities and focus on a small, but promising, region using a sequence of successively higher fidelities. Due to the increase of the number of dimensions and in cardinality $|\mathscr{X}|$ the performance of GP-UCB is reduced. To overcome this issue and control the discretization error (the error resulting from the fact that a function of a continuous variable is represented in the computer by a finite number of evaluations), it has been proved that GP-UCB can be run using the settings for finite set (Srinivas et al., 2012). An improved version of the GP-UCB is also suggested by Contal & Vayatis (2016) where, to precisely control the discretization error, a sequence of uniform discretizations are constructed using generic chaining that leads to tight bounds.

Following the GP-UCB algorithm (Srinivas et al., 2010), different sequential optimization schemes have been proposed in the literature incorporating the parallel strategy: multiple evaluations are performed in parallel whereby a batch of multiple input points is selected at each iteration. A batch optimization strategy was first introduced by Azimi et al. (2010), using the Monte-Carlo as an alternative to GP, based on the idea of simulation matching (SM). The proposed algorithm, SM-UCB, selects a batch of input points that closely match their expected behaviour. Choos-

ing the level of parallelism and whether to sequentially evaluate the function or evaluating it on a batch mode, was studied by Azimi et al. (2012). Specifically, in early stages, the algorithm chooses a new input point one by one and then naturally transits to a batch model where the batch size is adaptively changing based on the expected prediction error.

The connection between the multi-armed bandit and the experimental design was introduced by Srinivas et al. (2010) and extended by Desautels et al. (2014). Precisely, two algorithms accommodate the parallel strategy and the batch execution. The GP-BUCB, which selects at each iteration a batch of fixed size, and GP-AUCB, a variant of the first algorithm, which adaptively exploits parallelism to choose a batch of input points. The size of the batch in the GP-AUCB algorithm is based on the amount of information gained about the unknown function. The cumulative regret bounds are provided for both algorithms. Daxberger & Low (2017) generalizes the GP-BUCB algorithm by presenting DB-GP-UCB, a novel distributed batch optimization scheme which can jointly optimize a batch of inputs, as opposed to selecting the input points of a batch one at a time, and still preserve scalability in the batch size.

By combining two strategies to determine the input points for each batch of a fixed size, the GP-UCB-PE algorithm, proposed by Contal et al. (2013), aims to maximize an unknown function with the lowest possible number of function evaluations. Specifically, the UCB policy is used to select the first input point by balancing the exploration and exploitation whereas the remaining input points are chosen using the Pure Exploration (PE) strategy from regions which contain the true optimum with high probability. The PE step follows a greedy strategy where the input points are chosen one by one based on the ALM, the experimental design proposed by MacKay (1992). As a result, only the input points that maximize the information gain quantity, as in the work of Srinivas et al. (2010) and Desautels et al. (2014), are chosen. Under the GP framework, the information gain quantity

is computed based on the predictive variance.

Chapter 3

optim-MICE: A sequential global optimization algorithm

3.1 Problem statement and objectives

The current study addresses the problem of sequentially optimizing an unknown function. Let $f : \mathscr{X} \to \mathbb{R}$ be our unknown function with $\mathscr{X} \subseteq \mathbb{R}^d$, compact and convex. The aim is to find with the lowest possible number of function evaluations, the maximum of the unknown function

$$f(x^*) = \max_{x \in \mathscr{X}} f(x), \tag{3.1}$$

where x^* denotes the true location of the maximum of f. At each iteration t, a batch of K input points (x_t^k) in $\mathscr{X}$ are chosen and then the function values at these locations are simultaneously obtained.

The unknown function is modelled as a sample from a GP, with mean function $m(x)$ and covariance function $c(x;x')$. In the current study, the mean function is specified as zero whereas the covariance function is chosen from the *Matérn* family. Specifically, the smoothness parameter is set as $\nu = 5/2$. The hyper-parameters considered in the statistical surrogate model - GP regression - are estimated using

the approach proposed by Sacks et al. (1989). The correlation length for each input is estimated using the MLE approach.

The GP optimization is performed under the multi-armed bandit setting. We sequentially optimize the unknown function using the exploration vs. exploitation trade-off strategy: we explore the uncertain regions and exploit the available information to reach the supposed location of the maximum. Following the parallel strategy, at each iteration t a batch of input points K are chosen and the unknown function is evaluated simultaneously. The standard objective is to minimize the batch cumulative regret R_T^K,

$$R_T^K = \sum_{t<T} f(x^*) - f(x_t^k). \tag{3.2}$$

A number of objectives need to be achieved during the optimization procedure such as to find the value x^* that maximizes the unknown function in the lowest possible number of iterations or get a solution close to the true optimum, minimize the cumulative regret and balance the exploration vs. exploitation trade-off. Since the function f is an unknown and a computationally expensive black-box function, it is also important to be able to gain as much as possible information without necessarily performing a lot of function evaluations. This is mainly done using the MICE criterion. The unknown function f is evaluated only at inputs points which give the maximum information.

3.2 Parallel optimization procedure

The newly developed sequential surrogate-based optimisation scheme follows the two-step approach of a Bayesian Optimisation framework even if a Bayesian approach is not fully adopted. Firstly, the Gaussian Process regression, which is the probabilistic model, is built and then, the Upper Confidence Bound (UCB) is used as the acquisition function, to determine at which points the black-box function will

be evaluated next. At each step of the optimization process, the GP is refined by considering the new input points observed $\{x_t^k\}_{1\leq k<K}$, which are chosen in batches of a fixed size K. As in GP-UCB-PE (Contal et al., 2013), the first point is chosen based on the UCB policy, whereas the $K-1$ remaining points are chosen via the Pure Exploration strategy.

This section gives an overview of the optimization scheme and how the new input points are chosen at each step. It also shows the technical connection between the multi-armed bandit and experimental design, as first shown in Srinivas et al. (2012) and later in Contal et al. (2013). A simple example of the optim-MICE is illustrated in Fig. 3.1.

3.2.1 Confidence region

Under the GP framework, the predictive distribution at any input point x_t is again a multivariate Gaussian distribution, $GP(\hat{y}(x_t), \hat{s}^2(x_t))$ (as given in (2.15), (2.16)). Using this property, we can define a confidence region in which the unknown function f is included with high probability. The confidence bounds are constructed as in the GP-UCB acquisition function (2.30) and defined as:

$$\begin{aligned} \hat{f}_t^+(x) &= \hat{y}_t(x) + \sqrt{\beta_t}\hat{s}_t(x) \\ \hat{f}_t^-(x) &= \hat{y}_t(x) - \sqrt{\beta_t}\hat{s}_t(x), \end{aligned} \tag{3.3}$$

where f_t^+ is the upper bound, f_t^- is the lower bound, $\hat{y}(x_t)$ is the predictive mean and $\hat{s}^2(x_t)$ is the predictive standard deviation. The width of the confidence region is regulated by the value of the trade-off parameter β_t. It controls the exploration and exploitation, namely the balance between exploring the regions with high uncertainty (regions with high predictive variance) and focusing on the supposed input point of the maximum (input point that might give the highest reward).

The first input point, x_t^1, of each batch, is chosen based upon the UCB policy

and it is the one that maximizes the upper bound, or the GP-UCB acquisition function,

$$x_t^1 = \underset{x \in \mathcal{X}}{\arg\max}\, \hat{f}_t^+(x). \tag{3.4}$$

The upper and lower bounds as well as the first input point are shown in Fig. 3.1. The grey area is bounded by $\hat{f}_t^+$ and $\hat{f}_t^-$.

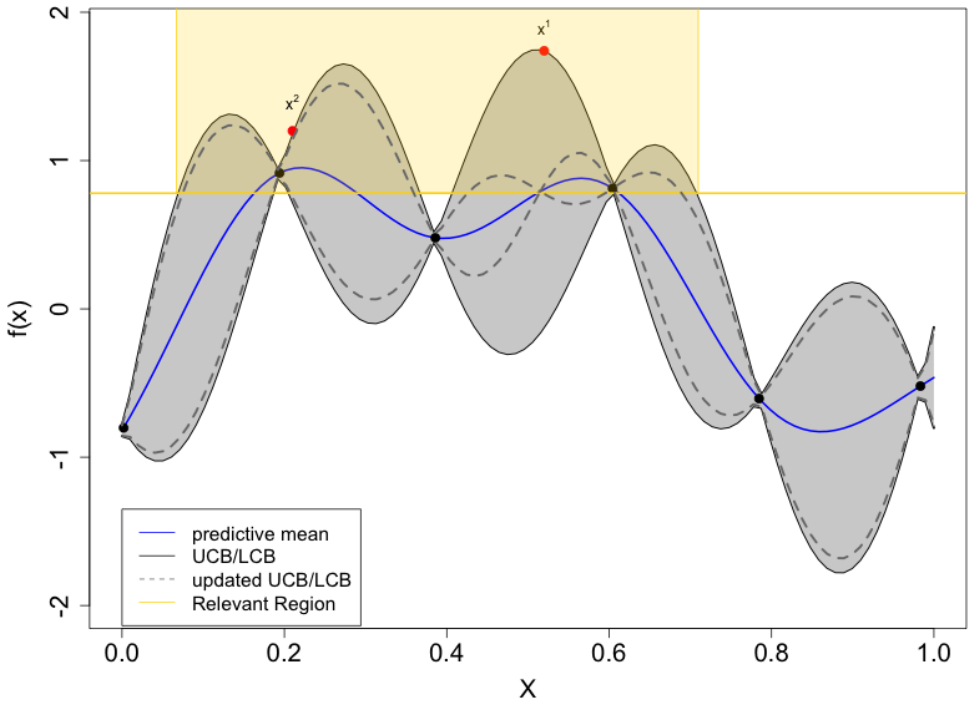


Figure 3.1: Illustration of optim-MICE applied in 1-dimensional test function. The grey area shows the confidence region and is bounded by $\hat{f}_t^+$ and $\hat{f}_t^-$. The first input point, x^1, is chosen based on the UCB-policy. The $y_t^\bullet$ is represented by the horizontal yellow line whereas the relevant region, $\mathfrak{R}_t$, is the yellow area. The black dashed lines shows the updated upper and lower bounds after having selected x^1. The second input point x^2 is chosen using the Pure Exploration strategy.

3.2.2 Relevant region

Having now specified the region which contains the unknown function f with high probability, a further reduction of that region is obtained where the true location of the maximum, x^*, of f belongs with high probability. The relevant region, $\mathfrak{R}_t$, is defined as

$$\mathfrak{R}_t = \left\{ x \in \mathscr{X} \mid \hat{f}_t^+(x) \geq y_t^\bullet \right\}, \tag{3.5}$$

where $y_t^\bullet$ is the lower confidence bound on the maximum, $y_t^\bullet = \hat{f}_t^-(x^\bullet)$ and $x_t^\bullet = \arg\max_{x \in \mathscr{X}} \hat{f}_t^-(x)$. At every iteration t, only the locations that might contain the true optimum of the unknown function f with high probability are kept in the relevant region. In Fig. 3.1, the $y_t^\bullet$ and the $\mathfrak{R}_t$ are represented by the yellow area.

3.2.3 Parallel evaluations using Mutual Information

Applying the parallel strategy, we are able to choose a batch of K input points at each iteration t. The $K-1$ remaining input points are selected via Pure Exploration. We restrict our attention to the relevant region $\mathfrak{R}_t$. The objective at this step is to maximize the information gain about the unknown function by selecting the most appropriate/informative input points.

In order to choose the next input point x_t^2, we calculate the MICE criterion (2.26) for all the input points available for selection. The point that is selected and added in the batch is the one that maximizes the MICE criterion. For all $1 < k < K$, using a greedy strategy, the new input points are selected one by one,

$$x_t^k = \arg\max_{x \in \mathfrak{R}_t} \hat{s}_t^2(x) / \hat{s}^2_{G \setminus (t \cup x)}(x; \tau_s^2), \tag{3.6}$$

where $\hat{s}_t^2$ is the updated variance after the next input point is chosen and included in the batch. The predictive variance does not depend on the unknown function

evaluation but only on the actual location of the next input point x_t^k. After choosing the $K-1$ input points, the uncertainty about the unknown function is reduced and the guess about the upper bound in the next iteration is improved. The second input point x^2 is selected after the Upper and Lower Confidence bounds have been updated. In Fig. 3.1 the updated bounds are represented by the dashed lines. The overall procedure is shown in Algorithm 1.

Algorithm 1 optim-MICE

for $t = 1, \ldots, T$ **do**
  Compute $\hat{y}_t$ and $\hat{s}_t$ with Eq. (2.15) and (2.16)
  $x_t^1 = \arg\max_{x \in \mathscr{X}} \hat{f}_t^+(x)$
  Compute $\mathfrak{R}_t$ with Eq. (3.5)
  for $k = 2, \ldots, K$ **do**
    Compute $\hat{s}_t^k$ with Eq. (2.16)
    $x_t^k = \arg\max_{x \in \mathfrak{R}_t} \hat{s}_t^2(x) / \hat{s}^2_{G \setminus (t \cup x)}(x; \tau_s^2)$ (2.26)
  end for
end for

3.3 Computational experiments

The empirical performance of the proposed method is compared with the GP-UCB-PE algorithm (Contal et al., 2013) and the q-points EI (Chevalier & Ginsbourger, 2013; Ginsbourger et al., 2008), a multi-points criterion for parallel global optimization based on the well-known EGO, on different optimization test functions. In this study, the GP-UCB-PE and the q-points EI are referred to as UCB-ALM and qEGO, respectively. The test functions have been selected to cover different input dimensions (from 2 to 6), physical properties, and shapes (Jamil & Yang, 2013). Despite the fact that none of the test functions is expensive to evaluate, a meaningful study of the performance of the proposed algorithm can be conducted

assuming that these functions are computationally expensive. It is expected that the proposed algorithm will behave the same on truly expensive functions as in test functions with similar surfaces.

The computational efficiency of the optimization algorithms is measured by calculating the cumulative regret: the loss incurred at iteration t due to not knowing the input points where f is maximized beforehand. Since the goal is to find the maximum of an unknown function with the lowest possible number of function evaluations, the different optimization schemes are compared in terms of the mean number of function evaluations (over multiple trials) required to get a solution with relative error $< 5\%$ and $< 1\%$. The relative error is given by $|f_{best} - f^*|/f^*$, provided that $f^* \neq 0$, where f_{best} is the best solution obtained by an algorithm and f^* is the true optimum. To meet the relative error requirement of $< 5\%$ and $< 1\%$ the target values for each optimization test function are calculated and presented in Table 3.1. In cases where the global optimum is zero, the target values are calculated based on the range of all the possible function values ensuring that the relative error is met.

3.3.1 Experimental set-up

In reality, the computational resources are limited, each function evaluation is costly, and the true optimum of a black-box function is unknown. The total number of function evaluations is restricted needs to be set in advance since a stopping criterion is not used in this study. So in this study, the total number of function evaluations is chosen before the optimization process starts for all the computational experiments. This number is determined by the size of the initial design drawn ($Ninit$), the number of iterations (T) and the batch size (K). Since the number of evaluations is a comparison metric for the current study, and to keep consistency, all the algorithm settings are chosen to be the same for all optimization methods. Table 3.1 gives the basic information of the test functions used in the current study and summarizes the domain and target values for each one of them.

Table 3.1: Test functions for the computational experiments (E)

Test function	Label	Dim	Design space	Global optim	Target E <1%	Target E <5%
Branin	$E1$	2	$[-5,10]\times[0,15]$	-0.398	-0.402	-0.418
Griewank	$E2$	2	$[-600,600]^2$	0	-0.2	-0.9
Himmelblau	$E3$	2	$[-6,6]^2$	0	-0.2	-1
Hosaki	$E4$	2	$[0,10]^2$	2.3458	2.3223	2.2285
Michalewicz	$E5$	2	$[0,\pi]^2$	1.8013	1.783	1.711
Sasena	$E6$	2	$[0,5]^2$	1.457	1.442	1.384
Six-Hump Camel	$E7$	2	$[-3,3]\times[-2,2]$	1.302	1.289	1.223
Zakharov	$E8$	2	$[-5,10]^2$	0	-0.05	-0.25
Harmann-3	$E9$	3	$[0,1]^3$	3.863	3.824	3.669
Rosenbrock	$E10$	3	$[-5,10]^3$	0	-1.8	-9
Powell	$E11$	4	$[-4,5]^4$	0	-1	-5
Sphere	$E12$	4	$[-5.12,5.12]^4$	0	-0.1	-0.5
Styblinski-Tang	$E13$	4	$[-5,5]^4$	156.664	155.097	148.831
Michalewicz	$E14$	5	$[0,\pi]^5$	4.688	4.641	4.453
Hartmann-6	$E15$	6	$[0,1]^6$	3.322	3.264	3.131
Trid	$E16$	6	$[-36,36]^6$	50	49.5	47.5

The initial input points are sampled using a maximin-distance design LHD. Regardless of the number of dimensions, the optimization procedure for each experiment starts with an initial design of $Ninit = 2$ input points. One may increase the size of $Ninit$ so as to cover better the input space with more points and possibly find the true optimum in fewer function evaluations. However, the effectiveness of using an adaptive design will not fully be achieved and this might lead to evaluate the unknown function over regions where uncertainty is low.

During the optimization process, $T \times K$ input points can be sequentially added in the design and therefore, in addition to the $Ninit$ runs, a further $T \times K$ evaluations can be performed. If the size of the resulting design is not restricted then choosing a bigger batch size gives the opportunity to explore more the uncertain regions without spending computational time re-estimating the GP parameters, but this increases the number of times the costly function is evaluated. In the current

study, the batch size is kept fixed for all the optimization methods, regardless of the dimensions: at each time t, a batch of $K = 5$ input points are selected. The number of iterations T is changed according to the number of dimensions: the higher the dimensions, the higher the number of iterations is.

The optimization results are affected by the size of the candidate set. Having a large number of candidate points increases the chances to end up with more input points in the Relevant Region ($\mathfrak{R}_t$) and be closer to the true optimum but, it also increases the computational time needed to examine all the candidate points regardless of design criterion. On the other hand, with a small candidate set, we might need more function evaluations to find the true optimum and might not fully benefit from the parallel exploration as the number of candidate points available for selection in the Relevant Region might be less than the predefined batch size K and therefore, the knowledge that we could obtain at each iteration for the unknown function will be minimum.

Due to the limited computational budget, only a certain number of candidate points can be examined using the MICE criterion. As a result, to cover the whole search space and ensure that enough candidate points have been placed in the important regions, 10^4 input points are initially sample using LHD where only a subset of them, which is randomly chosen, is examined with the MICE criterion. For convenience, the number of input points in the search space is defined as $Nsearch = 10^4$ whereas the candidate points available for selection $Ncand$. At each time step t, a new search set of size $Nsearch = 10^4$ is sampled and therefore, after the locations where x^* does not belong with high probability are discarded, a new set with a number of candidate points $Ncand$ is chosen for the PE. The number of candidate points, $Ncand$, is fixed at each iteration. For computational experiments with higher dimension, the size of the candidate set is chosen to be bigger.

For the UCB-ALM, all the input points included in the search set ($Nsearch = 10^4$)

are examined with the ALM criterion because its computational cost is low. Therefore, the number of candidate points available for selection is $Ncand = 10^4$.

The qEGO method is performed in R using DiceKriging and DiceOptim, two packages which are built for the approximation and the optimization of black-box functions and include the q-EI criterion (Roustant et al., 2012). Specifically, at each iteration, a batch of input points is obtained by maximizing the multipoint EI criterion using a hybrid genetic algorithm. To overcome multimodality of the EI function, and keep accuracy, the default settings are kept, which does not allow us to specify the values of *Nsearch* or *Ncand*. Furthermore, in the qEGO method, a GP model can only be fitted when the number of initial design points is at least one input point bigger than the number of dimensions. As result, the *Ninit* for the qEGO method is not kept fixed for all the computational experiments and is increased accordingly.

To get an accurate measure of the performance of all the optimization schemes, 50 trials of each of these algorithms on each test function are performed where each trial uses a different random seed. Table 3.2 gives a summary of the algorithm settings used for each optimization method.

Table 3.2: Algorithm settings

Algorithm Settings (No.)	Optimization methods	Dimensions				
		2D	3D	4D	5D	6D
Initial points	UCB-ALM, optim-MICE	2	2	2	2	2
	qEGO	3	4	5	6	7
Iterations		20	30	40	50	60
Batch size		5	5	5	5	5
Search space points	UCB-ALM, optim-MICE	10^4	10^4	10^4	10^4	10^4
	qEGO	-	-	-	-	-
Candidate points	UCB-ALM	10^4	10^4	10^4	10^4	10^4
	optim-MICE	50	100	150	200	250
	qEGO	-	-	-	-	-

3.4 Results and discussion

3.4.1 Computational efficiency

We compare optim-MICE with alternatives over 16 different computational experiments. Having performed 50 runs for each different experiment, a comprehensive picture of the computational efficiency of each optimization scheme is obtained. A summary of the results are shown in box-plots in Fig. 3.2a and 3.2b . The mean number of function evaluations required by each algorithm to get a solution with a relative error $< 5\%$ and $< 1\%$ is shown in Table 3.3. The mean function evaluations is calculated based only on the trials that achieved the target values. The number of successful trials (trials that achieved or exceed the target values) is given in the brackets. If none of the trials succeed to get an appropriate solution this is recorded as zero (shown in the brackets). Then, the mean function evaluations is recorded as a number greater than the total function evaluation specified to perform.

The 2-dimensional experiments examined are W-shaped ($E4$, $E6$), steep ridges ($E5$), valley-shaped ($E7$), plate-shaped ($E8$) with either one global maximum or more ($E1$) and sometimes with many local maxima ($E2$). Regardless of the surface and the complexity of the function, optim-MICE and UCB-ALM are doing definitely better than qEGO as all the best solutions obtained in each trial (Fig. 3.2a and 3.2b) are closer to the true optimum. For $E1$, $E3$, $E5$, $E6$ and $E7$, the target values and the true optimum in all the 50 trials performed are fully achieved (Table 3.3). For $E2$, $E4$ and $E8$ not all the trials are successful as some of them do not get a solution within 1% and/or 5% of the global maximum. However, the mean best solution, for both ALM- and MICE-based algorithms, always give a value close to the true optimum.

On the other hand, considering the spread of the solutions obtained in qEGO and the small number of the successful trials, the behaviour of algorithm can be unstable and the optimum achieved can be far away from the targets. A substantial

Table 3.3: Mean function evaluations of the trials achieved the target values with a relative error $< 1\%$ and $< 5\%$. The brackets show the number of trials that achieved the target values (out of the 50 performed). Zero in the bracket indicates that none of the trials succeed to get the target values.

E	Mean Function Evaluations $E < 1\%$			Mean Function Evaluations $E < 5\%$		
	UCB-ALM	optim-MICE	qEGO	UCB-ALM	optim-MICE	qEGO
$E1$	52(50)	**49(50)**	100+(0)	41(50)	**39(50)**	100+(0)
$E2$	64(12)	**50(13)**	100+(0)	23(50)	**21(50)**	8(6)
$E3$	44(50)	**39(50)**	71(3)	32(50)	**30(50)**	73(11)
$E4$	89(4)	**71(9)**	49(4)	61(29)	**57(41)**	39(7)
$E5$	59(50)	**58(50)**	67(3)	55(50)	**53(50)**	57(5)
$E6$	75(50)	**70(50)**	100+(0)	**52(50)**	57(50)	38(3)
$E7$	57(50)	**51(50)**	79(5)	44(50)	**40(50)**	47(11)
$E8$	79(11)	**78(26)**	45(4)	74(37)	**67(42)**	41(11)
$E9$	45(50)	**35(50)**	150+(0)	**28(50)**	35(50)	150+(0)
$E10$	120(21)	**116(29)**	150+(0)	104(50)	**100(50)**	150+(0)
$E11$	200+(0)	**197(13)**	200+(0)	191(7)	183(17)	**151(23)**
$E12$	52(31)	**45(34)**	200+(0)	29(50)	**26(50)**	167(15)
$E13$	103(36)	**97(39)**	200+(0)	76(50)	**70(50)**	200+(0)
$E14$	250+(0)	250+(0)	250+(0)	250+(0)	250+(0)	250+(0)
$E15$	193(3)	**179(9)**	300+(0)	88(50)	**82(49)**	300+(0)
$E16$	300+(0)	**119(8)**	300+(0)	107(14)	**102(11)**	300+(0)

Numbers in bold indicate the best result achieved in terms of mean function evaluations, among the three optimization schemes. In $E4$ and $E8$, optim-MICE performs better because the target values achieved in more trials.

difference between UCB-ALM and optim-MICE can be noted on the mean function evaluations required to meet the target values. In most cases, using optim-MICE, the target values can be achieved faster with fewer function evaluations, which imparts confidence in the computational efficiency of MICE.

In the higher dimensional experiments, the computational complexity increases and the target values are harder to achieve. Again, using optim-MICE surpasses the other two approaches, as for all the experiments ($E9$-$E16$) the number of successful trials is greater. To get a solution with a relative error $< 5\%$ and $< 1\%$, with optim-MICE, requires less function evaluations and therefore less computational

resources. For example, a solution with a relative error $< 1\%$ for the $E9$ and $E15$ is achieved by evaluating the function 10 and 14 fewer times, compared to the other algorithms, respectively (Table 3.3).

The summary results for the experiments $E9$, $E13$, $E14$ and $E15$ presented in Fig. 3.2a and 3.2b suggest that optim-MICE outperforms competing methods. The exact shape of $E9$ and $E15$ (Hartmann function) is unknown but it is well-known that it is a relatively smooth function with very few modes. The mean best solution for the 3- and 6-dimensional cases indicates UCB-ALM and optim-MICE are doing better than qEGO. A difficult test case is $E14$ (5-dimensional Michalewicz function) because part of the surface is plateau, which often hampers the search process of the optimization algorithm as these areas do not offer any information, and has steep ridges. The complexity of the function does not seem to be a problem in the 2-dimensional case ($E5$) but here, all the algorithms struggle to achieve the target values and find the global maximum. Comparing the three methods, optim-MICE does significantly better than the alternatives as, in most of the 50 trials performed, the best solution achieved is closer to the true optimum.

The global optimum of $E10$ (Rosenbrock) is located in a long narrow valley which makes the exploration process of an optimization algorithm even slower. Using either the ALM or MICE criterion, it is ensured to achieve the target values and a solution closer to the true optimum. As in $E7$, which is also a valley-shaped function, so too in $E10$, the function values obtained with qEGO are far away from the true optimum indicating its difficulty to find the uncertain region.

For most of the global optimization methods relied on heuristic techniques, $E11$ (Powell) is an easy problem (Steihaug & Suleiman, 2013) whereas for other methods, including the proposed algorithm, it is a challenging test case. Considering the summary results in Fig. 3.2b, optim-MICE and qEGO are significantly better than UCB-ALM. What attracts the attention here is that with qEGO the mean function

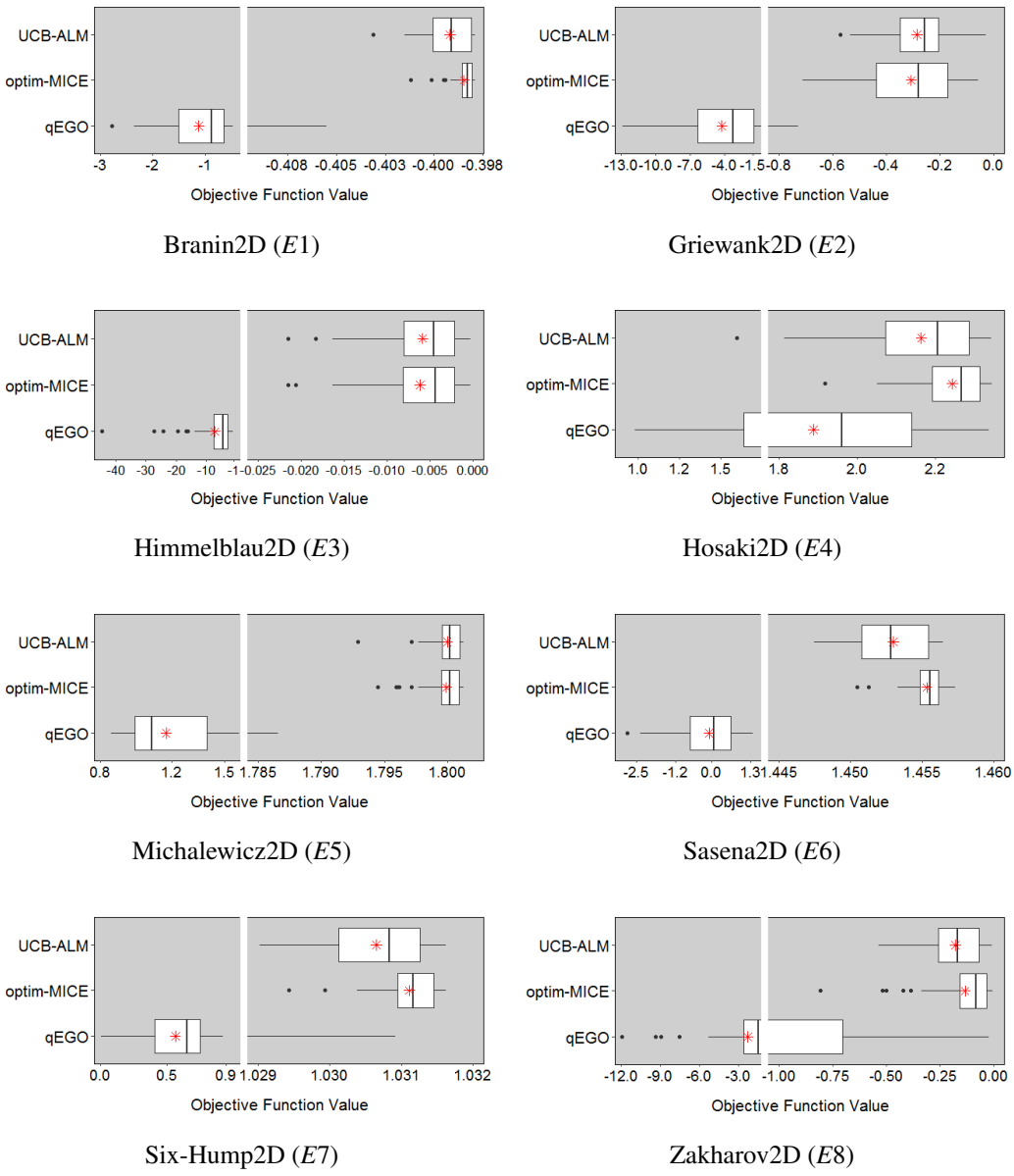


Figure 3.2a: Summary of the best solution achieved in the 50 trials in box-plots with a gap in the range of function values. Red star shows the mean best solution.

evaluations required to get a solution within 5% of the true optimum is much lower, and the number of successful trials is higher, than the alternatives but none of the trials performed gets a solution with a relative error $< 1\%$. On the other hand, with optim-MICE, the best solution achieved is even closer to the true optimum.

Both functions, $E12$ (Sphere 4D) and $E16$ (Trid 6D), have a bowl-shaped surface

but, based on their 2-dimensional illustrations, the uncertain region of $E12$ is in the middle of the surface while Trid's global maximum is positioned towards the edge. In contrast with UCB-ALM, optim-MICE spots the uncertain region of $E12$ faster and the mean function evaluations required to achieve the target values is smaller. This is also in line with the structure of the MICE criterion which, at each time step, forces us to choose a new point that yields the least uncertainty between itself and the unselected input points. This point is usually 'central' with respect to points that have not been selected yet (Krause et al., 2008). As ALM tends to place many points in the boundaries of the design space, it is expected to find the uncertain region faster and perform better in $E16$. The number of trials that found a function value within 5% of the true optimum is higher than in optim-MICE but none of the trials got a solution with a relative error $< 1\%$ (Table 3.3). Considering the range of the best solutions achieved in the 50 trials (Fig. 3.2b), it can be stated that the overall performance of optim-MICE is better.

3.4.2 Exploring Branin's input space: the visited locations

The Branin function has been selected to examine further as it is a 2-dimensional case with three global maxima. Therefore, this function is a challenging case as we need to explore the entire search space and identify the three different uncertain regions. A contour plot for the Branin function and the three global maxima can be seen in Fig 3.3. Fig. 3.4 shows all the design points chosen for $E1$ (Branin), from each algorithm, of the trial with the best (left) and worst (right) solution. Green triangles show the three global maxima whereas red ones show the optimal solution obtained from each algorithm. Choosing the design points based on the MICE criterion proves to be advantageous as in both trials, best and worst, the uncertain regions are fully explored without spending computational time on unnecessary areas. Although with the ALM criterion the uncertain regions are also successfully spotted on both trials, more computational time is spent exploring unimportant regions and evaluating the objective function on locations far away from the true optimum. It also tends to push a number of design points near the boundaries which

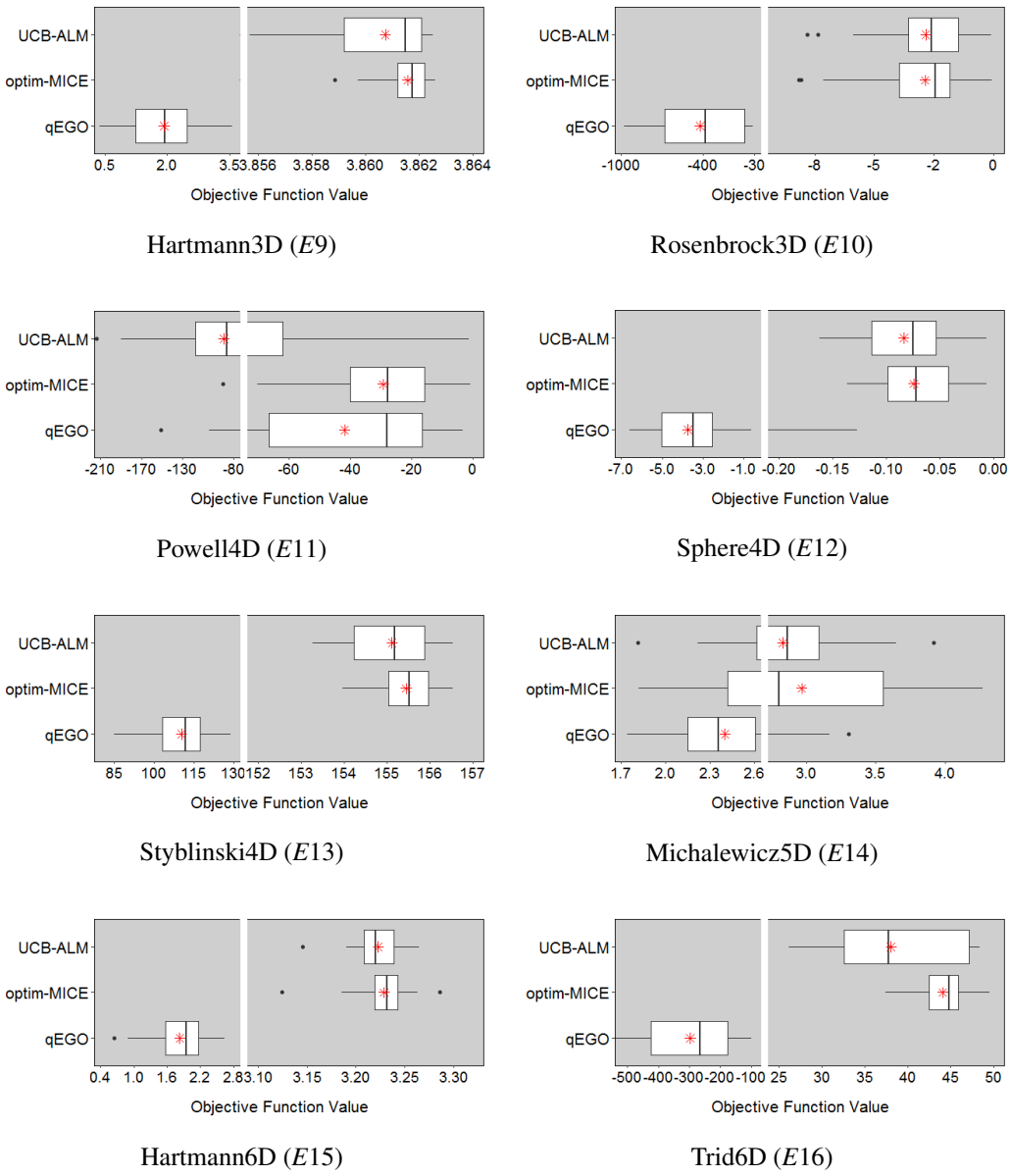


Figure 3.2b: Summary of the best solution achieved in the 50 trials in box-plots with a gap in the range of function values. Red star shows the mean best solution.

might make it attractive for cases where the optimum is in the edges.

Considering the input points chosen with qEGO, the visited locations are spread around the entire design space. As in the best trial, so too to the worst trial, it lacks the ability to identify the important regions, and stick to them, which results in a bad design set and often the true optimum is not found within the predefined

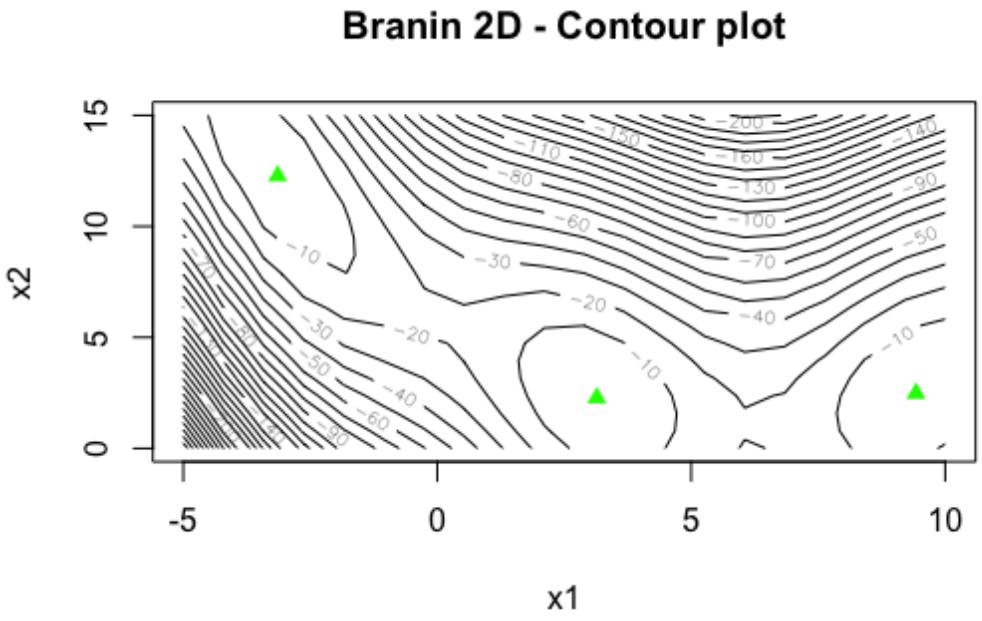


Figure 3.3: Contour plot of the Branin function. Triangles show the three global maxima.

time horizon. The information gained about the objective function from the chosen points is limited, and a valuable computational time is wasted on evaluating it on unhelpful locations. A possible reason that might affect the overall performance of qEGO is the small number of initial points: at the initial stage of the optimization, the knowledge is minimal and the right direction is difficult to be found.

The way that qEGO explores the input space looks to be beneficial when the true optimum is not isolated in a small region of the search space. Such examples are $E4$ and $E8$, where the uncertain region covers a huge area and is not interrupted by ridges or drops. Despite the fact that qEGO could not achieve the target values in most of the trials performed in $E4$ or $E8$, and the mean best solution is far from the true optimum (Fig. 3.2a), for the ones that perform well, the average function evaluations required to a get a solution with a relative error $< 5\%$ and $< 1\%$ is much lower compared with the other methods. By slightly increasing the number of initial design, and the corresponding initial computational cost, qEGO could possibly perform better in some particular experiments.

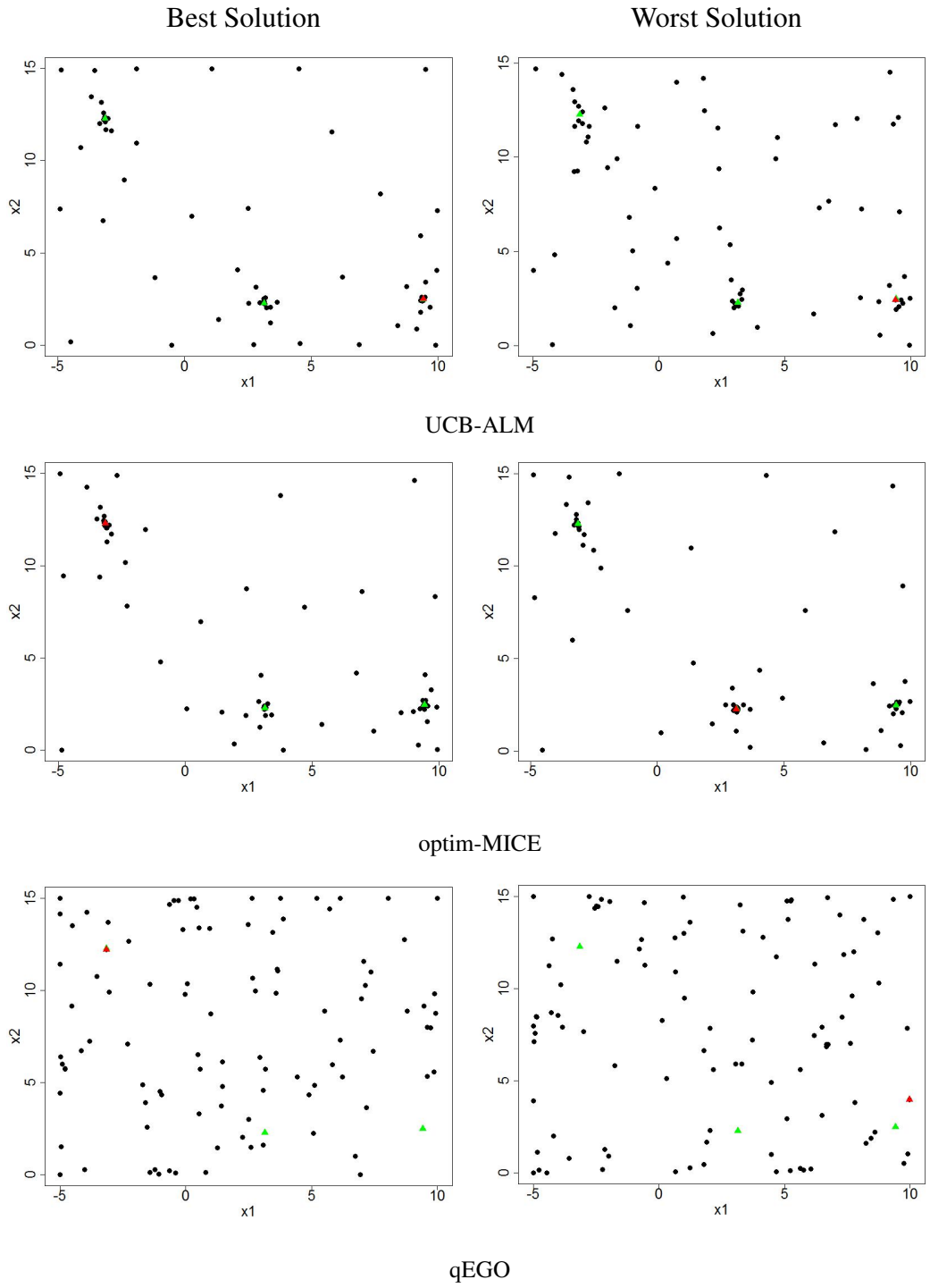


Figure 3.4: Design points selected from each optimization approach for the Branin function ($E1$) according to the best (*left*) and worst (*right*) solution achieved among the 50 trials. Green triangles show the three global maxima and red triangles show the optimal solution obtained from each algorithm

3.4.3 The convergence of the regret

The computational efficiency of the three algorithms is further examined by calculating the simple regret. Fig. 3.5a and 3.5b show the evolution of the mean simple regret for nine of the experiments. The small plots are zoomed around the most interesting part. The mean simple regret is calculated based on the true optimum and the mean current solution taken of the 50 runs.

In more details, with ALM- and MICE-based criterion, the mean regret in $E1$ converges to zero after the same number of function evaluations, whereas with qEGO, this does not seem happening in the specified time horizon. This is also explained by the fact that none of the 50 trials performed achieved a solution close to the true optimum (see Table 3.3). A difficult case for all the algorithms, as it is mentioned earlier, is when the objective function has many local maxima. Such a case is the $E2$ where, with qEGO, even if a lot of function evaluations are performed, the regret is not converging to zero. In contrast, the decay of regret is done faster in UCB-ALM and optim-MICE but, to converge to zero and get a solution literally close to the global maximum both need a lot more function evaluations.

The efficiency of the search process followed by the MICE-based algorithm is also noted in $E7$, $E8$, $E13$ and $E16$ as the convergence is achieved in fewer function evaluations than in alternatives. In terms of qEGO, except in $E9$ where it seems to struggle a lot, in all the other experiments the regret decreases, but slowly, and does not converge to zero within the predefined time period common to all methods. As it is expected, despite the improvement seen in $E14$ during the optimization procedure, the complexity level of the function affects the algorithm's search process. To achieve convergence with any of the three methods, more function evaluations would be required, and tuning the algorithm settings would be also helpful. Overall, considering the fast decay of the regret, it can be stated that optim-MICE finds the uncertain region and gets a solution close to the true optimum in less computational time, regardless of dimensionality and complexity of the function.

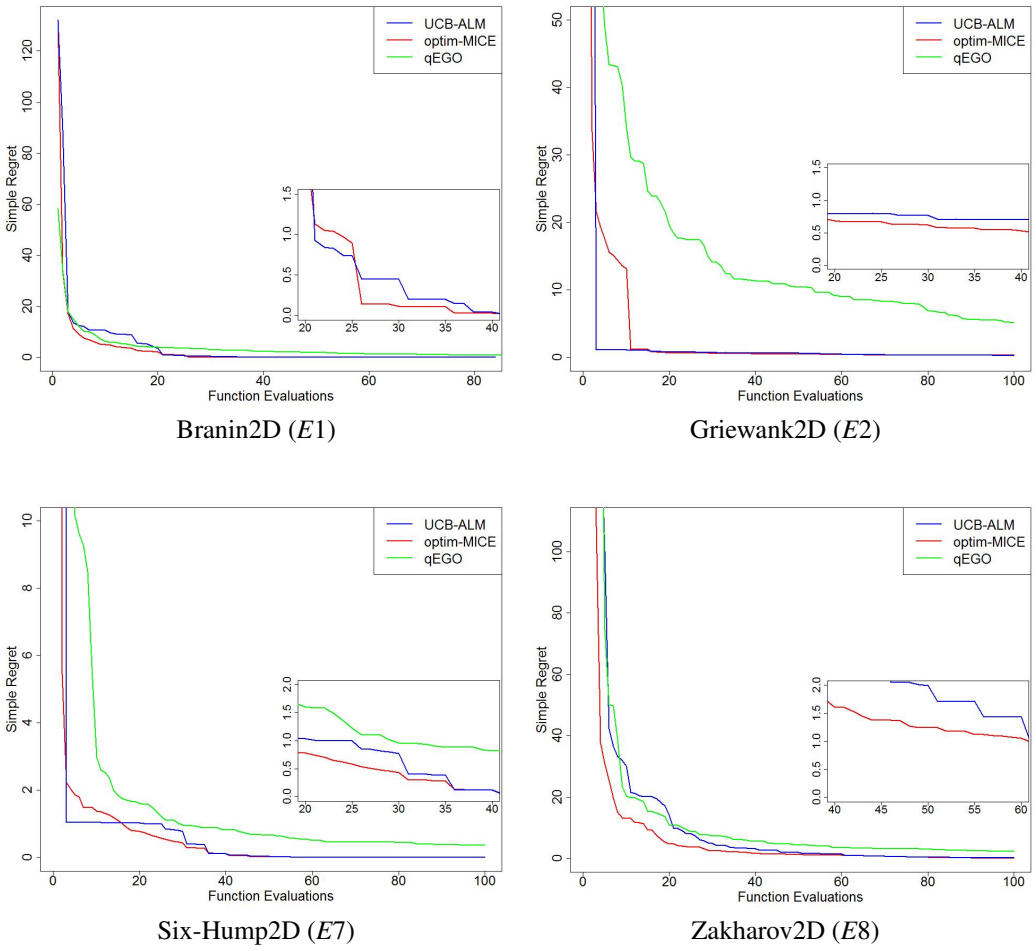
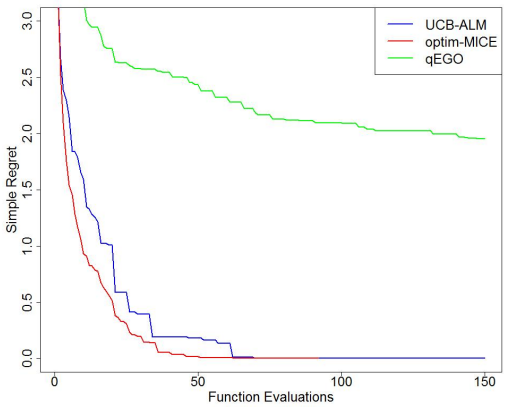
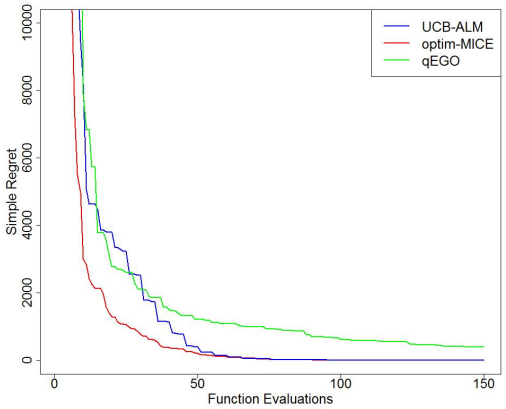


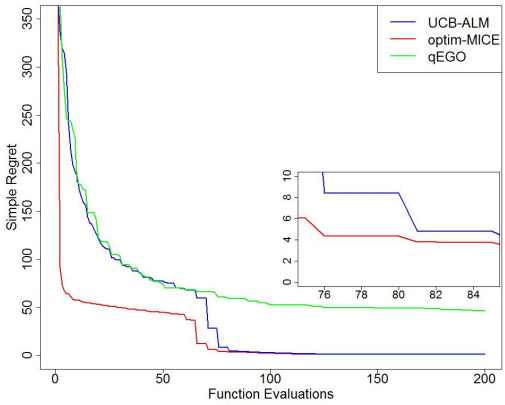
Figure 3.5a: Comparison of the mean simple regret with respect to the total function evaluations performed during the optimization process. Small plots show a zoomed part of the decay of the regret.



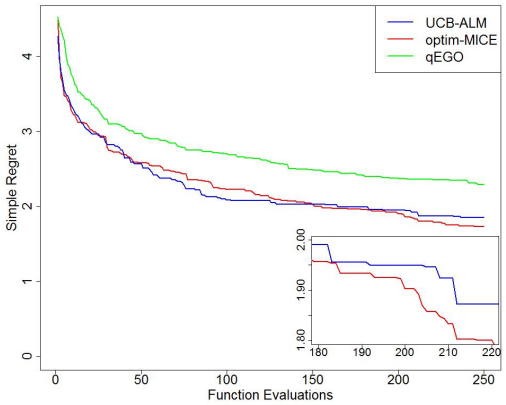
Hartman3D ($E9$)



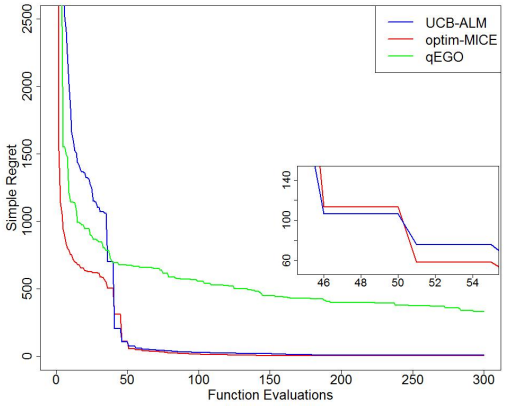
Rosenbrock3D ($E10$)



Styblinski4D ($E13$)



Michalewicz5D ($E14$)



Trid6D ($E16$)

Figure 3.5b: Comparison of the mean simple regret with respect to the total function evaluations performed during the optimization process. Small plots show a zoomed part of the decay of the regret.

Chapter 4

Tuning settings

The main settings of optim-MICE are the number of iterations (T), the batch size (K), the number of input points in the search space available for selection (*Nsearch*) and the number of candidate points ready to be examined with MICE (*Ncand*). To understand their effect on the proposed algorithm, a simple sensitivity analysis is performed on four of the experiments: Hosaki 2D ($E4$), Sasena 2D ($E6$), Rosenbrock 3D ($E10$) and Hartmann 6D ($E15$). Specifically, one setting is varied while others remain fixed. In total, 12 different scenarios are presented and compared over each test function. The base scenario has the same algorithm settings as in the previous chapter (Table 3.2). As before, 50 trials performed for each scenario.

The effect of the main settings on the overall performance of the algorithm, and whether its computational efficiency is improved, are also examined in a scaled version of $f(x)$. Different transformations of the objective function are compared under the 12 scenarios. At two different levels, Hosaki 2D ($E4$) and Rosenbrock 3D ($E10$) are both scaled vertically and horizontally by varying one algorithmic setting at a time and compared with the non-scaled version. Each scenario is performed 50 times. A vertical scaling multiplies or divides every function value by a constant while leaving the input variables unchanged. In contrast, a horizontal scaling multiplies or divides every input variable by a constant while leaving the function value unchanged. In theory, scaling the objective function during an optimization

study does not matter as the location of the maximum within the input space stays the same. However, it is expected to have a significant influence on the overall performance of the optimization algorithm as the appearance of the search space and the uncertain region changes accordingly i.e sometimes they stretch and some others shrink.

Tables 4.2 and 4.4 show the best solution achieved, the mean best solution and standard deviation as well as the average number of function evaluation required to get a solution with a relative error $< 5\%$ for $E6$ and $E15$, respectively. Tables 4.1 and 4.3 show the average number of function evaluations required to achieve the target values for $E4$ and $E10$ under different scaled versions. The first row in all the four tables (4.1 - 4.4) shows the algorithm settings used for the base scenario and its results. The results of the 50 trials performed for each scaled version of the objective function are summarized in box-plots (Fig. 4.5 and 4.6). The computational efficiency of the algorithm is also measured by calculating the cumulative regret for all the different scenarios for each test function (see Fig. 4.1-4.4). The simple regret is calculated based on the true optimum and the mean current solution taken of the 50 runs. All the tables show the number of successful trials - those which achieved the target value - in the brackets. If the target value is achieved in none of the trials, then this is recorded as zero.

4.1 Algorithm settings: effect and importance

By increasing the number of iterations, the mean best solution is closer to the true optimum, the standard deviation, as expected, becomes smaller and more trials succeed to get a solution within the acceptable level of error. But does the overall performance get better? Clearly, a small number of iterations can affect the precision of the results and the solution obtained can be far from the true global maximum (e.g. in $E4$, $E6$ and $E10$). Considering the mean simple regret, the convergence to zero can be achieved but this happens only in a later stage as it requires

more computational time. If the computational time is not a constraint, a higher number of iterations can be beneficial and add value to the entire optimization process, but not always. For example, in $E4$ and $E10$, the more iterations performed the better the mean best solution is, whereas in $E6$ and $E15$, after a certain number of iterations there is no substantial improvement and a lot of computational time is wasted.

Table 4.1: Tunning Algorithm Settings: Scaling Hosaki 2D ($E4$). Brackets show the number of successful trials, out of the 50 performed, which achieved a solution with a relative error $< 5\%$.

T	K	Nsearch	Ncand	Mean Function Evaluations (E <5%)				
				Non-Scaled Version	Scaled Versions			
					Vertical		Horizontal	
					$0.5f(x)$	$2f(x)$	$f(0.5x)$	$f(2x)$
20	5	10^4	50	57(41)	55(44)	57(44)	60(41)	59(42)
10	5	10^4	50	28(12)	22(12)	22(10)	29(8)	29(7)
30	5	10^4	50	84(47)	80(45)	81(47)	82(44)	83(46)
50	5	10^4	50	83(50)	75(50)	77(50)	84(50)	85(50)
20	**2**	10^4	50	30(6)	18(5)	20(6)	21(4)	23(8)
20	**10**	10^4	50	98(48)	94(48)	97(49)	104(48)	107(50)
20	**15**	10^4	50	92(50)	89(50)	92(50)	102(50)	105(50)
20	5	**50**	50	58(26)	58(35)	58(36)	58(33)	58(33)
20	5	$\mathbf{10^3}$	50	58(30)	57(39)	57(32)	61(32)	63(31)
20	5	$\mathbf{10^5}$	50	54(41)	51(43)	53(40)	55(36)	55(40)
20	5	10^4	**25**	61(27)	56(27)	55(29)	60(31)	60(31)
20	50	10^4	**100**	50(44)	51(47)	51(45)	56(29)	60(31)

Numbers in bold indicate the algorithm setting that is varied in each different scenario.
Grey-coloured column shows the best scaled version of $f(x)$

The amount of exploration needed is not known beforehand but it can be controlled by the size of the batch. Using a small batch size, the uncertain regions are not fully explored and the mean best solution is far from the optimum (e.g. $E4$, $E10$). In this case, there is a high possibility to fail to achieve the target value (e.g. when $K = 2$, Tables 4-7), or to perform a lot of function evaluations until the decay of the regret is achieved or even fail to convergence into zero. On the other hand, a large batch size helps the algorithm in the search process in the very first few iterations,

where the knowledge about the unknown function is limited, and always ensures a good solution. As more input points are chosen in each iteration, the regret decays straight away indicating that the uncertain region can be found in few function evaluations (Fig. 4.1 and 4.3).

But having a larger batch size does not always guarantee a massive improvement of the overall performance of the optimization. For example, in $E4$ and $E10$, a larger batch size gets a solution even closer to the true optimum but not as close as if more iterations would be performed, in $E6$, a batch of 10 or 15 input points does not make substantial difference whereas in $E15$, a batch of 5, 10 or 15 input points yields almost the same mean best solution. Once the uncertain region is found, the progression can be slow and the convergence to zero can only be achieved after a lot of function evaluations (Fig. 4.2 and 4.4). A possible reason might be that the explored region is tiny and any new input point selected for the design will add little or no information about the unknown function as the candidate points drawn in that region will almost be identical. That slows down the performance of the algorithm and makes it much harder to see an improvement.

What it is also worth mentioning is that the number of function evaluations required to achieve the target value is increased as the batch size is increased and, that number is much bigger if it is compared with the scenarios of increasing the number of iterations. This indicates that exploring the search space more than is needed is not always computationally efficient, despite the better solution that can be achieved at the end.

Although optim-MICE is computationally more expensive than the other two alternative schemes, it is more effective. The quality of the points added in the design at each time step is better and as a result, the true optimum is achieved in less function evaluations. Due to the limited computational resources, the MICE criterion is only examined on a certain number of candidate points. If the number of potential design

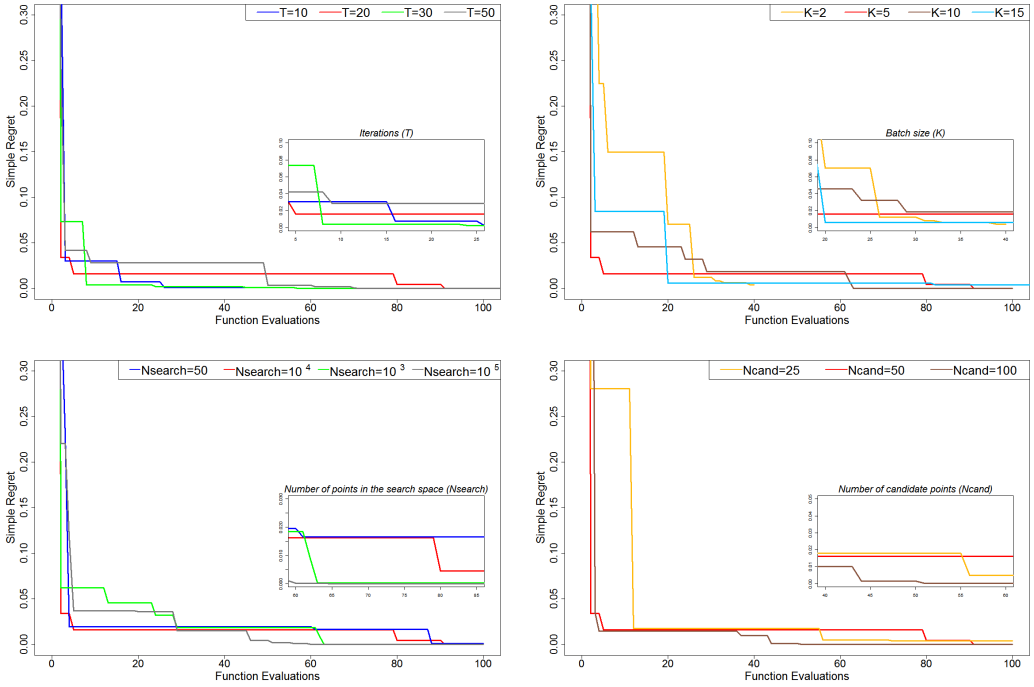


Figure 4.1: Mean simple regret for Hosaki 2D ($E4$) under different combinations of the algorithm settings. *top*: Number of iterations (T) and Batch size (K). *bottom*: Number of points in the search space (*Nsearch*) and Number of candidate points (*Ncand*).

points included in the search space is small, and the same as the number of candidate points (e.g. $Nsearch = 50$ and $Ncand = 50$, Table 4.1) which are used for the exploration part, the uncertain regions are not fully explored. The solution is often far from the true optimum and the regret converges to zero after a lot of function evaluations. Considering the results obtained in the four experiments, where only half of the 50 trials performed are successful, there is a also a higher possibility to fail to achieve a solution with a relative error $< 5\%$. In contrast, as it is expected, having a high *Nsearch*, the target value is found in less computational time and the decay of regret is done faster.

The number of candidate points examined by MICE is a vital setting to the overall performance of the algorithm. Despite the fact that in the ALM-based algorithm a much larger number of candidate points are examined at each iteration, the results in the computational experiments show that the MICE-based algorithm still

Table 4.2: Tunning Algorithm Settings: Sasena 2D ($E6$). Brackets show the number of successful trials, out of the 50 performed, which achieved a solution with a relative error $< 5\%$.

T	K	Nsearch	Ncand	Best Solution	Mean Best Solution	SD	Mean Function Evaluation $E < 5\%$
20	5	10^4	50	1.4564	1.4553	0.0012	57(50)
10	5	10^4	50	1.4523	1.0126	0.5270	37(9)
30	5	10^4	50	1.4565	1.4566	0.0024	58(50)
50	5	10^4	50	1.4565	1.4559	0.0031	58(50)
20	**2**	10^4	50	1.4366	0.9441	0.7010	29(7)
20	**10**	10^4	50	1.4565	1.4561	0.0004	65(50)
20	**15**	10^4	50	1.4565	1.4561	0.0003	68(50)
20	5	**50**	50	1.4556	1.1801	0.8431	60(21)
20	5	$\mathbf{10^3}$	50	1.4560	1.3630	0.6103	55(47)
20	5	$\mathbf{10^5}$	50	1.4565	1.4555	0.0012	62(50)
20	5	10^4	**25**	1.4564	1.4551	0.0067	61(50)
20	5	10^4	**100**	1.4565	1.4540	0.0019	50(50)

Numbers in bold indicate the algorithm setting that is varied in each different scenario.

performs better even with a smaller candidate set. Considering the results obtained in the four test cases, a larger set of candidate points gives a solution even closer to the true optimum with the lowest possible number of function evaluations. The proposed algorithm also performs well with a small candidate set as the mean best solution is still within an acceptable distance from the true optimum. However, with a larger set, the average number of function evaluation required to get a solution with a relative error $< 5\%$ is much lower (Tables 4.1 - 4.4). How fast the regret decays depends on the complexity of the function, but having a large number of candidate points allows the regret to converge in fewer function evaluations (e.g. Fig. 4.1 and 4.2).

4.2 Stretching and shrinking the uncertain region

Compared with the non-scaled version, scaling $E4$ either vertically or horizontally does not make a substantial difference in the solution achieved. Overall, the best

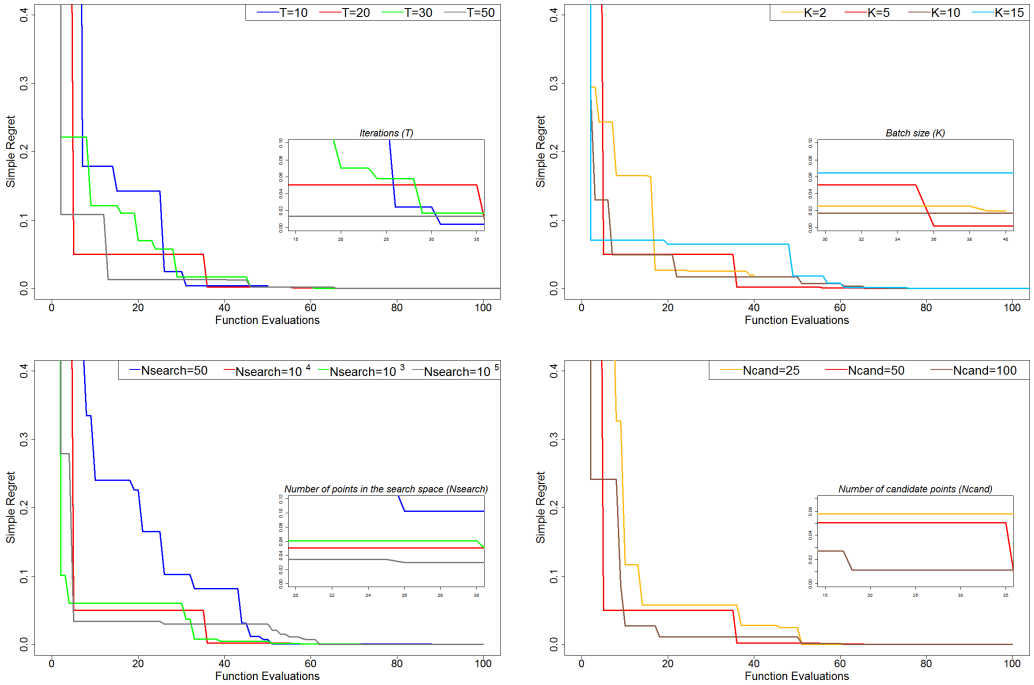


Figure 4.2: Mean simple regret for Sasena 2D (*E*6) under different combinations of the algorithm settings. *top*: Number of iterations (T) and Batch size (K). *bottom*: Number of points in the search space (*Nsearch*) and Number of candidate points (*Ncand*).

solutions achieved from all the 50 trials in each scaled-version, shown in box-plots in Fig. 4.5, are very close to the solutions achieved in the non-scaled version. In contrast, the mean function evaluations required to get a solution with a relative error $< 5\%$, when the objective function is scaled vertically, is minimized even more compared with the non-scaled and the horizontal scaled versions.

Hosaki (*E*4) is a function with two isolated regions - one large and one small - and a huge uncertain region which lies in the large one. Multiplying the input variables by a scale factor less than one, the entire uncertain region stretches in the horizontal direction and that makes the search space even wider. On the other hand, by multiplying the input variables by a scale factor greater than one causes all the input variables to increase and the uncertain region to shrink. Compared to $y = f(0.5x)$, this horizontal transformation is more effective as the best solutions achieved, in most of the scenarios, are closer to the true optimum. However, in both

Table 4.3: Tunning Algorithm Settings: Scaling Rosenbrock 3D ($E10$). Brackets show the number of successful trials, out of the 50 performed, which achieved a solution with a relative error $< 5\%$.

T	K	Nsearch	Ncand	Mean Function Evaluations (E <5%)				
				Non-Scaled Version	Scaled Versions			
					Vertical		Horizontal	
					$0.5f(x)$	$2f(x)$	$f(0.5x)$	$f(2x)$
30	5	10^4	100	100(50)	75(50)	112(42)	88(50)	97(41)
20	5	10^4	100	83(31)	74(45)	84(29)	78(34)	73(34)
40	5	10^4	100	94(50)	78(50)	121(50)	87(50)	99(50)
60	5	10^4	100	99(50)	81(50)	118(50)	86(50)	100(50)
20	**2**	10^4	100	47(5)	53(11)	46(2)	54(5)	43(4)
20	**10**	10^4	100	114(50)	82(50)	138(50)	102(50)	118(50)
20	**15**	10^4	100	115(50)	97(50)	159(50)	104(50)	137(50)
20	5	**100**	100	94(12)	81(29)	98(4)	79(15)	82(18)
20	5	$\mathbf{10^3}$	100	97(41)	87(49)	112(26)	98(45)	92(41)
20	5	$\mathbf{10^5}$	100	96(50)	79(50)	114(48)	85(50)	101(49)
20	5	10^4	**50**	95(50)	83(50)	114(35)	90(50)	99(48)
20	50	10^4	**150**	95(50)	79(50)	113(42)	79(50)	105(47)

Numbers in bold indicate the algorithm setting that is varied in each different scenario.
Grey-coloured column shows the best scaled version of $f(x)$

horizontal scaled-versions, the number of function evaluations required to get the target value is higher than in the non-scaled one. This happens possibly because the true optimum ends up, in both scaled-versions, in an area where the algorithm requires more function evaluations in order to get the target value i.e. if the uncertain region is stretched a lot the true optimum ends up in a flatted area whereas if it shrinks more than what is needed, the true optimum lies in a very tiny area.

Among the scaled-versions, vertical scaling ($y = 0.5f(x)$ and $y = 2f(x)$) seems to perform better than horizontal scaling. Considering all the trials performed, most of the best solutions achieved in each different scenario are closer to the true optimum. As in the non-scaled version, in the scaled ones, by increasing the number of iterations and the batch size and, choosing the design points from a larger candidate set, the computational efficiency is improved. Although by scaling vertically the objective function by a factor greater than one (e.g. $y = 2f(x)$) seems to be bene-

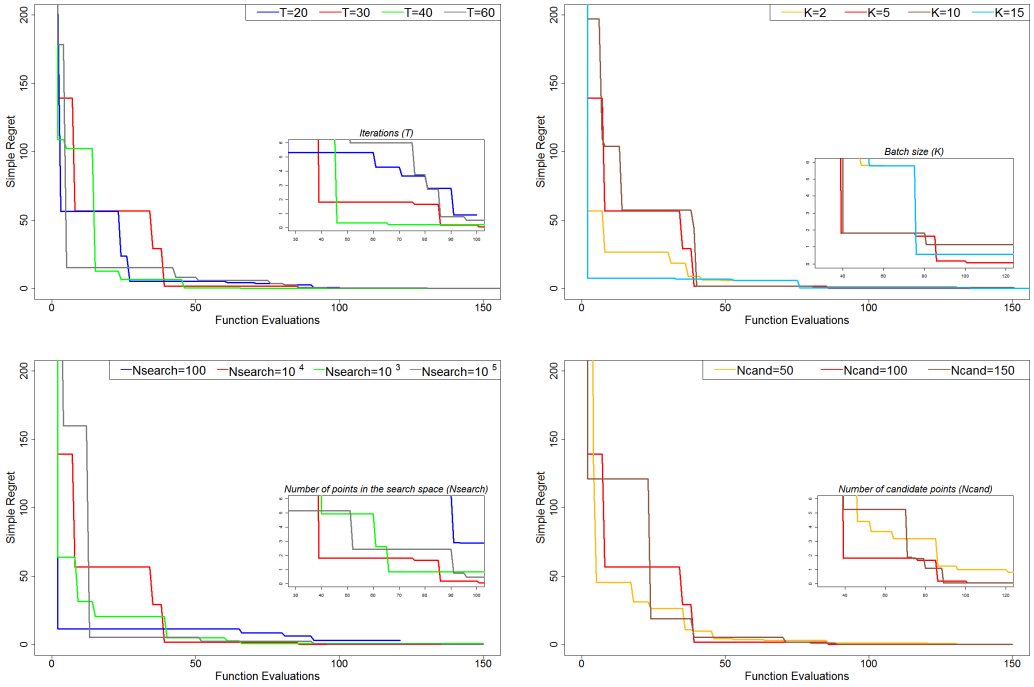


Figure 4.3: Mean simple regret for Rosenbrock 3D ($E10$) under different combinations of the algorithm settings. *top*: Number of iterations (T) and Batch size (K). *bottom*: Number of points in the search space (*Nsearch*) and Number of candidate points (*Ncand*).

ficial and overall a good solution can be achieved, the $0.5f(x)$-case is even more computationally efficient. What is really achieved by scaling vertically the objective function by a factor of 0.5, is that the mean function evaluations required to get a solution with relative error $< 5\%$ is lower than in the other examined versions (Table 4.1).

By multiplying the objective function by a value less than one causes all the function values to decrease and the uncertain region to shrink in the vertical direction. With this transformation, not only we can get a solution close to the true optimum but also we can find the best solution in the lowest possible function evaluations. This mainly happens because the interval of the possible function values is also shrunk, while the search space remains unchanged, and any new input point added in the design during the exploration part is very likely to be close to the true optimum. Furthermore, by exploring a shrunken uncertain region in the vertical direction the

information obtained is more valuable. Due to that, the knowledge about the objective function is soon increased during the optimization procedure and the target value can be achieved in less computational time.

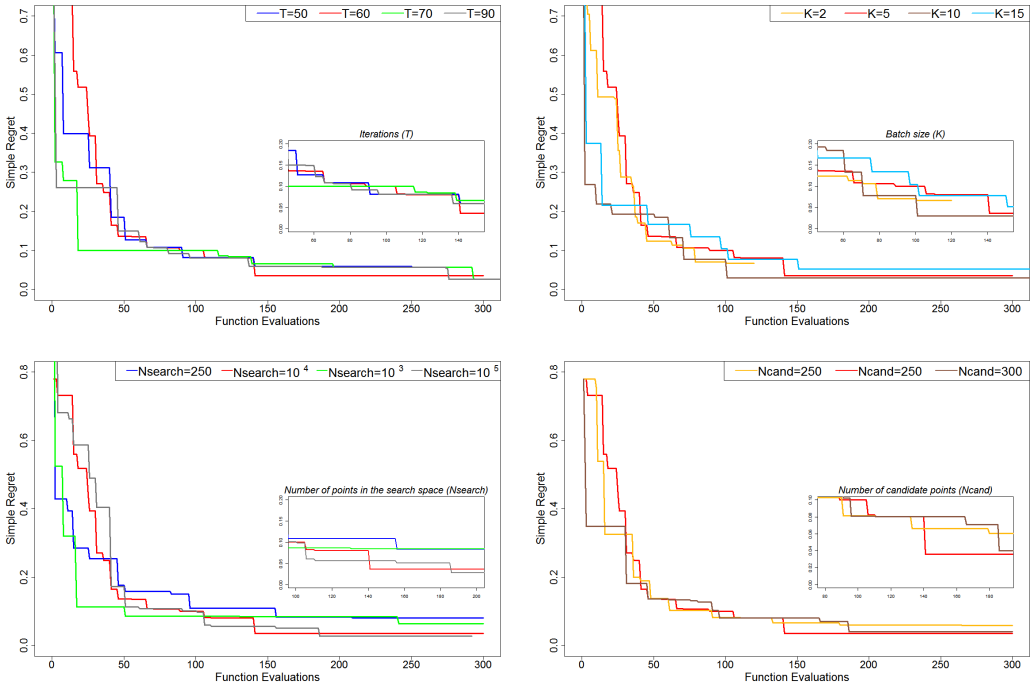


Figure 4.4: Mean simple regret for Hartmann 6D ($E15$) under different combinations of the algorithm settings. *top*: Number of iterations (T) and Batch size (K). *bottom*: Number of points in the search space (*Nsearch*) and Number of candidate points (*Ncand*).

The advantage of scaling the objective function is better shown in $E10$. Rosenbrock ($E10$) is viewed as a difficult function to be optimized as the global maximum, in its 2-dimensional case, resides inside a long narrow and parabolic-shaped valley. The problem becomes much harder in the higher dimensional cases and to find the uncertain region and get the true optimum is trivial. However, using optim-MICE and scale the objective function vertically by 0.5 seems to be beneficial as the overall performance of the algorithm is improved. The results are shown in Fig 4.6. For convenience, we plot the negative logarithm of function values i.e if f is the function values, we plot the $-\log(-f)$.

Table 4.4: Tunning Algorithm Settings: Hartman 6D ($E15$). Brackets show the number of successful trials, out of the 50 performed, which achieved a solution with a relative error $< 5\%$.

T	K	Nsearch	Ncand	Best Solution	Mean Best Solution	SD	Mean Function Evaluation $E < 5\%$
60	5	10^4	250	3.286	3.228	0.023	106(49)
50	5	10^4	250	3.260	3.191	0.031	114(49)
70	5	10^4	250	3.296	3.229	0.019	117(50)
90	5	10^4	250	3.296	3.235	0.014	113(50)
60	**2**	10^4	250	3.250	3.188	0.058	72(41)
60	**10**	10^4	250	3.292	3.231	0.021	143(50)
60	**15**	10^4	250	3.292	3.230	0.020	170(50)
60	5	**250**	250	3.241	3.158	0.036	156(24)
60	5	$\mathbf{10^3}$	250	3.259	3.189	0.031	130(40)
60	5	$\mathbf{10^5}$	250	3.294	3.258	0.011	86(50)
60	5	10^4	**200**	3.263	3.225	0.024	110(48)
60	5	10^4	**300**	3.285	3.230	0.022	101(50)

Number in bold indicate the algorithm setting that is varied in each different scenario.

Overall, by shrinking the uncertain region in the vertical direction the variation between the best solutions achieved in each trial is smaller than in the other examined versions (Fig 4.5 and 4.6). The number of successful trials is also bigger and that imparts even more certainty about the algorithm's performance. In all the different scenarios performed, the mean best solution is closer to the true optimum and the mean function evaluations required to get a solution with a relative error $< 5\%$ is much lower than in any other version.

Despite that the sensitivity analysis of the algorithm settings and the scaling method are done in four experiments, some general recommendations for the choice of the tuning settings and the scale factor can be given as the test functions used differ on the level of complexity and physical properties covering a huge variate of different situations. Some generic recommendations are explained in Chapter 6. It would be ideal to perform more experiments so as to get a better understanding of the

effect of the algorithm settings on the proposed algorithm. However, in reality, each computer model (true function) is different so it is hard to suggest specific recommendations for the choice of tuning parameters.

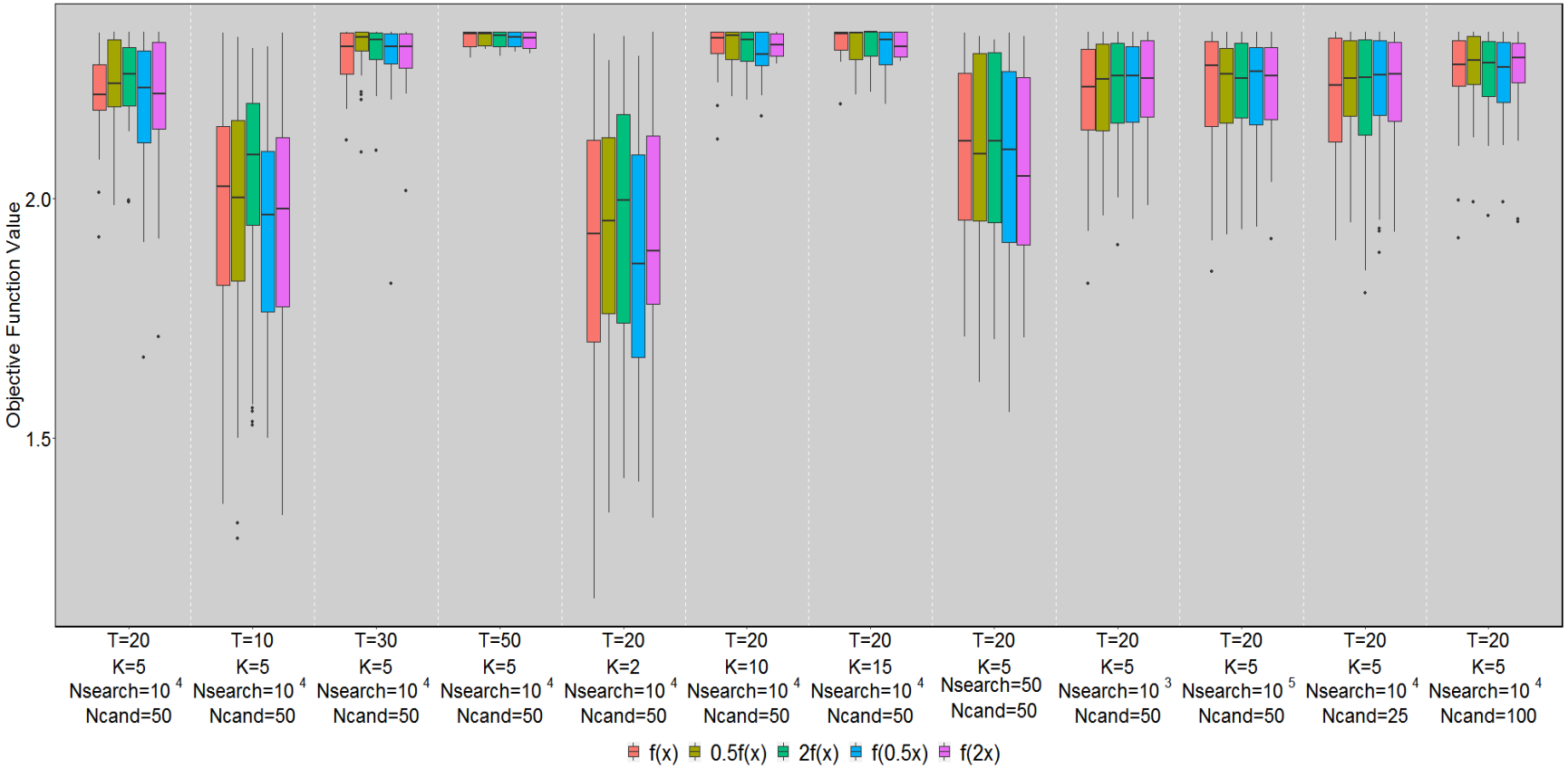


Figure 4.5: Summary of the best solution achieved in the 50 trials in box-plots for the scaled-versions of Hosaki 2D (*E*4)

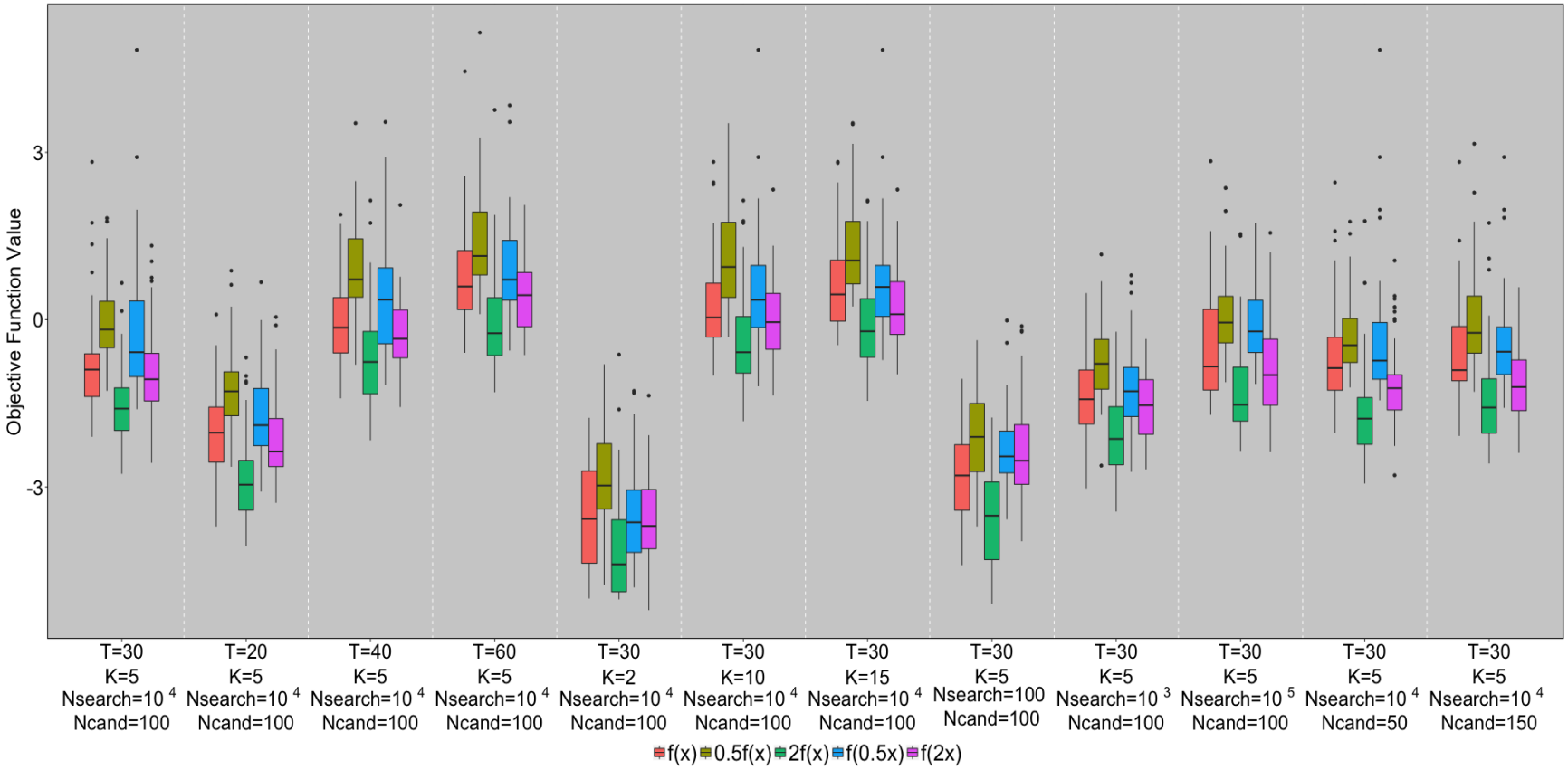


Figure 4.6: Summary of the best solution achieved in the 50 trials in box-plots for the scaled-versions of Rosenbrock 3D ($E10$). The y-axis shows the negative logarithm of the negative function values ($-\log(-f)$).

Chapter 5

Surrogate-based optimization of storm waves heights and run-ups

5.1 Background and aims

Storm surges cause coastal inundation due to setup of the water surface resulting from atmospheric pressure, surface winds and breaking waves. They occur during tropical cyclones, also known as typhoons or hurricanes, and are referred as the major geophysical risks associated with coastal areas due to the large numbers of casualties and damage (Ellis & Sherman, 2015; Feng et al., 2018).

Typhoons are subject to climate change-related influences, such as warmer sea surface temperatures and sea level rise (Bengtsson et al., 2009; Lin et al., 2012; Wang et al., 2015). Despite the fact that modelling storm surge events under different climate scenarios is not straightforward, changes in the storm surges' intensities and frequencies in a warmer climate have been investigated with high-resolution climate models (Bengtsson et al., 2009; Lin et al., 2012). However, different studies have shown that, statistically, weather-climatic extremes (e.g. wind, precipitation) are still sometimes over-or under-estimated due to the difficulty to evaluate them in climate simulations, adding another layer of uncertainties to assess future scenarios (Muis et al., 2016; Romero & Emanuel, 2017). Therefore, it is crucial to quantify

carefully the local impact of storm waves following these regional climate forcings in order to estimate future (and possibly increasing) storm wave risks.

About half of the strongest typhoons in the Western North Pacific hit the Philippines, as the country lies in the most tropical cyclone-prone waters on Earth (Ribera et al., 2008; Takagi et al., 2017). On 8 November 2013, one of the strongest tropical storm events ever recorded in the Philippines, Typhoon Haiyan, damaged billions worth of agriculture and infrastructure and, caused 6340 casualties. The town of Hernani, which was completely destroyed, got the attention of multiple research groups (Roeber & Bricker, 2015; May et al., 2015; Soria et al., 2018). What is interesting about the situation of Hernani, is that the broad fringing coral reef near its coast was expected to protect the coastal communities (Ferrario et al., 2014) however, in the case of Hernani, the reef exacerbated the damage from energetic waves.

Furthermore, as explained by Roeber & Bricker (2015), the nature of this event has remained a mystery for some time. Initially, it was assumed that the damage was caused by a storm surge, but the water level due to barometric-, wind-, and wave-induced set-up would not have carried the destructive power - especially because storm surges move rather slowly. The possibility of a meteo-tsunami or a near-field tsunami was also precluded as the propagation speed of the storm system was not in phase with the local wave celerity and no tsunami-generating earthquake or landslide activity was reported. For reference, meteo-tsunami are large waves driven by air-pressure disturbances often associated with fast-moving weather events such as severe thunderstorms whereas near-filed tsunami is triggered by seismic activity and consists of a series of waves rather than a single one, just like tsunamis, but occurs within 200 km of the epicentre of an earthquake and quickly arrives near the coasts (Roeber & Bricker, 2015).

Various studies state that the strong wind from Typhoon Haiyan pushed enough

water onshore causing a storm surge height of 6m (Mori et al., 2014). The areas most affected by the storm surge were close to Tacloban, where the bathymetric conditions favored the generation of such a massive storm surge. Surprisingly, destructive long-period bores struck the town of Hernani. Though the actual storm surge was rather small, wave run-up heights of over 7m have been reported around Hernani's Pacific coastline (Soria et al., 2016). The duration of the flooding caused by each bore was on the order of a minute and therefore much longer than the period of regular storm waves. The tsunami-like waves, which resulted from the wave-breaking process over the steep reef face, have been shown to be generated by the energetic surf beat under some particular conditions that favored the generation of infra-gravity waves (e.g. reef geometry, wave patters whose ranges of variations will depend upon climate forcing). Due to the fact that a similar tsunami-like flood was only reported during the 12 October 1897 typhoon, coastal risk assessment and evacuation plans did not consider the existence of this rare but disastrous event. The phenomenon in Hernani was successfully captured by Roeber & Bricker (2015) using a Boussinesq-type phase-resolving wave model, called BOSZ, able to reproduce the transfer of short wave energy into energetic long-period infra-gravity waves (Roeber & Cheung, 2012).

In general, the impact of inundation and coastal flooding events is often quantitatively assessed by the maximum wave run-up. The wave run-up is defined as the maximum vertical extent of a wave's up-rush on a beach or structure above a known reference level such as a chart datum or simply the mean water level. Understanding the conditions that caused the wave run-up and obtaining accurate predictions are of utter importance for coastal flooding hazard predictions. Due to that, wave run-up has been extensively studied either to observe the interaction between tsunami-like waves and fringing reefs (Yao et al., 2018; Ning et al., 2019), or to understand the influence of infragravity (long) waves (Shimozono et al., 2015; Montoya & Lynett, 2018), and the effects of climate change on coastal flooding risk (Lin et al., 2012; Quataert et al., 2015).

Various statistical approaches are also widely used to study the wave run-up. Extreme Value Theory (EVT) is applied to analyse the storm peak significant wave height and estimate the failure probabilities of offshore structures (Northrop et al., 2017). It is also used to evaluate the risk of storm tides considering projected changes to cyclone behaviour and the impact of wave setup (an effect of breaking waves at the coast) (McInnes et al., 2003). Hakkou et al. (2019) estimated the extreme total water level using classical EVT theory taking into account the differences of coastal morphology in order to assess the flooding. Using a point-wise and spatial statistical model to build a high-resolution hindcast data-set, Sartini et al. (2017) investigated the spatial variability of extreme significant wave heights aiming to improve the understanding of the processes governing wave climate. The combined impact of wave height and other variables (water level, wind waves etc.) has been studied by applying advanced probabilistic approaches (Jonathan et al., 2014; Leijala et al., 2018). Multivariate copula functions are also widely used to model the dependence between wave height, wave period, water level and storm duration and to analyse their extremes (De Waal & Van Gelder, 2005; Li et al., 2014; Rueda et al., 2016). The above studies are based on data-driven methods.

To overcome the lack of data and explore various scenarios, numerical models are widely used in this field. They provide valuable information about the coastal flooding events as they can compute the relevant physical processes efficiently and capture the extreme environmental events successfully. But because they are computationally demanding, further studies and more detailed analysis, such as sensitivity analysis, become impractical (Sarri et al., 2012). To overcome this issue, empirical equations are used as a simpler method to estimate run-up elevation during extreme surge events. Due to the fact that these methods are mostly derived from specific data-sets and represent a range of commonly encountered conditions, their applicability to more complex geophysical situations is limited and within their calculations can possibly include a substantial error (Park & Cox, 2016; Ji et al., 2018).

Recent studies highlight the need for further understanding the risk associated with natural hazards in order to improve coastal resilience (Behrens & Dias, 2015). The advancement in computational methods has given the opportunity to reduce the computational burden of numerical simulations and perform challenging studies such as the quantification of the uncertainty in the predictions of hazard characteristics. Statistical emulators, which have been shown to be a prominent solution and alleviate the computational cost issues, are starting to be used in this field.

Sarri et al. (2012) used a Gaussian Process statistical emulator to accurately approximate the landslide-generated tsunami model built by Sammarco & Renzi (2008) whereas Beck & Guillas (2016), proposed an improved experimental design to efficiently build an emulator for a tsunami model. Statistical emulators have been also used in more recently to quantify tsunami hazard over the North Atlantic Ocean (Salmanidou et al., 2017) and Cascadia (Guillas et al., 2018), to explore how eruption source parameters affect volcanic radiative forcing (Marshall et al., 2018), to quantify the uncertainty in Manning's friction parametrization applied to predict the sea surface elevations for the 2011 Japanese tsunami (Sraj et al., 2014), to identify different sources of uncertainty contributed in volcanic ash transport and dispersion simulators (Spiller et al., 2014), as well as to perform fast and efficient predictions of estuarine hydrodynamic variables, such as the water levels and non-tidal residual, in the USA Pacific Northwest (Parker et al., 2019).

A statistical emulator is often seen as an integral part in an optimization algorithm, which aims to maximize a numerical simulator, especially in situations where the traditional mathematical approach can not be taken due to the complexity of the simulator. Stefanakis et al. (2014) used an active experimental design strategy to find the combination of parameters which will give the maximum run-up amplification of typical tsunami waves with the lowest possible model runs. The method explores the entire input space and focuses on the uncertain region, which contains

the targeted location of the maximum, using the Active Learning MacKay (ALM), an active experimental design. At each time step, each combination of the input parameters is examined considering its predictive variance (a measure of uncertainty of the point predicted). The same optimization technique was integrated into a methodology of predicting and optimizing the layouts of wave energy converters in a wave farm (Sarkar et al., 2016).

The current study focuses on finding the worst case scenario from extreme coastal storm and infra-gravity waves - such as the ones that destroyed the town of Hernani - with a minimum computational effort. A suite of storm waves that lead to extreme wave run-up and bore heights are studied to understand sensitivities to inputs and to identify the conditions that will create possibly large storm wave run-ups at the local level. To do that, optim-MICE, the newly developed optimization scheme proposed in Chapter 3, is used to find the combination of reef and wave spectral parameters that affect surf beat run-up and forces on coastal structures. Taking advantage of the computational efficiency of the optim-MICE algorithm, the maximum run-up and bore height are estimated using lowest possible number of evaluations and thus keeping the computational requirements to the minimum. This is the first maximization of storm surge heights and run-ups using surrogate-based optimization.

5.2 Storm surge simulation set-up

The numerical simulations are performed using BOSZ (the Boussinesq Ocean and Surf Zone model), a Boussinesq-type phase-resolving wave model, which can handle the generation of large nearshore waves and their breaking process over an abrupt fringing reef (Roeber & Cheung, 2012). BOSZ has successfully captured the Typhoon Haiyan storm surge event, which happened back in Philippine Islands as it is able to compute the tsunami-like surf beat over the reef near Hernani. The massive bore that destroyed Hernani was an extreme infra-gravity surge, which resulted from the transfer of energy from short swell waves to long infra-gravity

waves initiated by wave breaking at the reef edge. The capability of the BOSZ model in handling these wave processes provides the baseline for studying different aspects of storm wave extrema such as the maximum bore height and the maximum run-up. The maximum bore height is defined as the highest free surface elevation over the section of the horizontally flat reef, whereas the maximum run-up is the highest vertical elevation on the beach, to which the waves reach. Both measures are governed by different processes. To study both extremes the model input needs to be optimized separately.

In practice, one of the challenges in an optimization task is to decide when to stop the iterative strategy. A number of different empirical stopping criteria have been proposed in the literature where the entire optimization procedure is stopped once a rule is satisfied (Azimi et al., 2012; Stefanakis et al., 2014). But in reality, the computational resources are limited and only a certain number of function evaluations can be performed. Due to the computational complexity of BOSZ, we fix the total number of function evaluations, and therefore the computational time, able to perform in both tasks. The predefined limit is in line with the algorithm settings chosen in Chapter 3 (Table 3.2) ensuring the computational efficiency of optim-MICE regardless of dimensionality and function complexity. For the Bore Height case, which is a four-dimensional, a total of 40 iterations are performed having a batch of size 5 and at each time-step 150 candidate points are available for selection. On the other hand, for the Run-Up case, which is a five-dimensional problem (as one of the physical parameters is proved to be not influential during the variable importance assessment, see Section 5.3.1), a total of 50 iterations are performed having a batch of size 5 and at each-time step 200 candidate points are available for selection.

The two main inputs of BOSZ are the bathymetry/topography and the offshore wave spectrum. For each extreme measure a different simplified bathymetric profile is created whereas the wave generation for the wave input is based on the

empirical JONSWAP distribution (Hasselmann et al., 1973), which depends solely on the values of significant wave height, Hs, and peak period, Tp. Tables 5.1 and 5.2 show the physical parameters incorporated in each optimization task. The range of the parameters was selected to account for reasonable and physically possible combinations. It was not the goal to utilize the maximum values of the parameters. Fig. 5.1 and 5.2 show the simplified bathymetric profiles used in the optimization task and illustrate the meanings of the physical parameters.

Table 5.1: Physical Parameters for Bore Height

Parameter	Measure	Range
alp: Fore-reef slope	degrees	3 - 33.33
h2: Water depth over the reef	m	0.01 - 3
Hs: Significant wave height	m	1 - 5
Tp: Peak Period	s	10 - 18

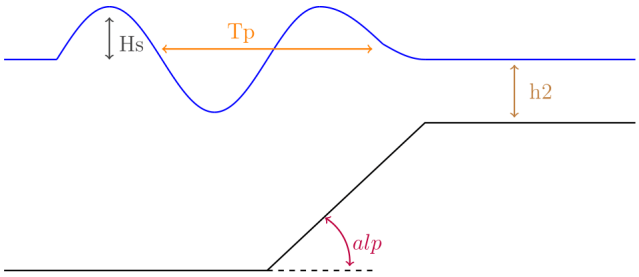


Figure 5.1: Bore height: Bathemetric profile of the experimental set-up.

Table 5.2: Physical Parameters for Run-Up

Parameter	Measure	Range
alp: Fore-reef slope	degrees	3 - 33.33
h1: Water depth offshore	m	50 - 75
h2: Water depth over the reef	m	0.01 - 3
bet: Beach slope	degrees	1 - 30
Hs: Significant wave height	m	1 - 5
Tp: Peak Period	s	10 - 18

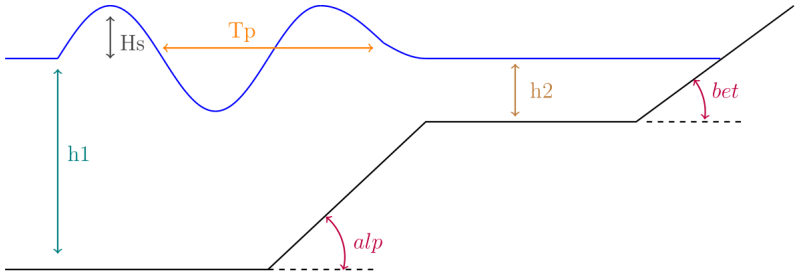


Figure 5.2: Wave Run-up: Bathemetric profile of the experimental set-up.

5.3 Storm surge extrema: results and discussion

BOSZ is a modular computer code, which in this study is treated as a black-box model. The computational expense of a Boussinesq-type model is relatively small in comparison to a full 3D model; however, thousands of individual runs can require substantial amounts of time even if the computation is executed only along a transect across the shoreline. The presented technique ensures that the variation of input variables efficiently influences the computer model's output in a way that the maximum bore or run-up heights are quickly achieved. The overall efficiency of the computer model is improved and a large amount of computational time is saved during the optimization procedure. To be more precise, the entire optimization task takes approximately 10 hours in total on a cluster of 4 cores whereas, each run of BOSZ takes about 2.5 minutes for our idealised set-up used here to demonstrate capabilities. For realistic geometries ideally modeled on a two-dimensional domain at high resolution each run could take even hours and the total computational time of performing such an optimization task would increase sharply.

5.3.1 Variable importance assessment

To find which input variables (physical parameters) are significantly contributing to the model output, before the optimization starts, we choose to perform the Morris Method (Morris, 1991), an initial Sensitivity Analysis (SA), also known as a Screening technique, which is widely used in the area of computer experiments (Boukouvalas et al., 2014; Iooss & Marrel, 2017) and its effectiveness and effi-

ciency has already been proved (Campolongo et al., 2007; Sanchez et al., 2014).

With a small number of model evaluations and without relying on strong assumptions about the computer model, the Morris method explores quickly the model's behaviour and ranks the input variables according to their impact on the model output. The input space for each input variable, as given in Tables 5.1 and 5.2, is discretized in levels and then, a factorial sampling strategy is used to construct trajectories where one input variable is varied at a time by a pre-defined step Δ, while keeping all others fixed. The sensitivity information is obtained by using the defined trajectories and computing for each input variable a number of incremental ratios, called Elementary Effect (EE), from which basic statistics are then calculated. EEs, which are approximations of the first order partial derivatives, are defined as the ratio between the variation in the model output and the variation in the input variable itself (Morris, 1991). The sensitivity measures obtained from EEs are then used to classify the input variables into groups based on their effects.

In more details, suppose that the model Y has d independent input variables, for any possible value of these d parameters in the input space, i.e., $\mathbf{X} = [X_1, X_2, \ldots, X_d]$, model output $Y(\mathbf{X})$ can be defined as $y(X_1, X_2, \ldots, X_d)$. Each input variable $\mathbf{X}_i$, $i = 1, \ldots, d$, is varied across p selected levels in the input space. Thus, the experimental region is discritized in a d-dimensional p-level grid. The input variables are assumed to be uniformly distributed in $[0, 1]$ and then transformed from the unit interval to their actual distributions.

Using the Morri's sampling strategy (Morris, 1991; Campolongo et al., 2007), a sequence of $d+1$ points are sampled in the grid, which is often called trajectory. The first point sampled is randomly selected whereas the next sample differs only in one coordinate from the preceding one. At each step, the ith input variable X_i is changed by a pre-defined step Δ, while all the other input variables remain unchanged. The direction to the new point and which input variable will be modified

are both randomly selected. The model output is computed for every point in the trajectory. Based on the trajectory obtained, the EE, known also as the coefficient of variation, is computed for each input variable. The EE of the ith input variable is defined as:

$$d_i(\mathbf{X}) = \frac{y(X_1,\ldots,X_{i-1},X_i+\Delta,X_{i+1},\ldots,X_d) - y(\mathbf{X})}{\Delta}, \tag{5.1}$$

where Δ is a multiple of $1/(p-1)$ and $\mathbf{X}(X_1,X_2,\ldots,X_d)$ is any selected value in the grid such that the transformed point $(\mathbf{X}+e_i\Delta)$ is still in the experimentation region for each index $i = 1,\ldots,d$ and e_i is a vector of zeros but a unit as its ith component.

To be able to evaluate the global sensitivity of the model and obtain a global measure considering the whole input space, a set of r different random trajectories of $d+1$ points needs to be constructed at the cost of $r \times (d+1)$ (Campolongo et al., 2011; Sanchez et al., 2014). Having r trajectories, a finite distribution of the EEs is formed which allows deriving statistical measures of the overall importance of each input variable. The various trajectories obtained differ by their starting point and the order of modified coordinates.

A challenging problem is to choose the number of trajectories r in order to obtain a good estimate of the EEs. Despite the fact that the analysis is more precise when r is high, it increases the computational cost significantly. The literature suggests either a number for trajectories or different approaches on how to optimally choose r. The current work uses the Optimal Trajectory (OT) approach where from a huge number of trajectories M which are built at the beginning, we select the combination of r trajectories with the highest "spread" in the experimental region using the Euclidean Distance (ED). It is a two-step approach: for each possible combination of r trajectories out of M, we first calculate the sum of all the distances, defined as D, between couples of trajectories. Then, the combination with the highest value of D is chosen. The distance ed_{ml} between a couple of trajectories,

m and l is defined as:

$$ed_{ml} = \begin{cases} \sum_{i=1}^{d+1} \sum_{j=1}^{d+1} \sqrt{\sum_{z=1}^{d} \left[X_i^m(z) - X_j^l(z) \right]^2}, & \text{for } m \neq l \\ 0, & \text{otherwise,} \end{cases} \tag{5.2}$$

where d is the number of input variables and $X_i^m(z)$ indicates the zth coordinate of the ith point of the mth Morris trajectory. More details can be found in Campolongo et al. (2007) but here is a simple example on how ed_{ml} is calculated as given in their work: for instance, we set $r = 4$ and select the combination 4, 5, 7 and 9 out of the possible $M = \{1,2,3,4,5,6,7,8,9,10\}$. The quantity $D_{4,5,7,9}$ is defined as $D_{4,5,7,9} = \sqrt{ed_{4,6}^2 + ed_{4,7}^2 + ed_{4,9}^2 + ed_{6,7}^2 + ed_{6,9}^2 + ed_{7,9}^2}$.

Once r trajectories are collected, common statistical measures used to evaluate the EEs. To express the sensitivity of the input variables, we calculate the mean μ and the standard deviation σ, both proposed by Morris (1991), and the absolute mean μ^* proposed by Campolongo et al. (2007). The main purpose of calculating the absolute mean μ^* is to ensure that all the effects will be considered in the variable importance assessment (e.g. if the model is non-monotonic, some effects may cancel each other out). In general, μ assesses the influence of each input variable on the model output and measures the overall sensitivity and the standard deviation σ, measures the involved interactions and non-linearity effects of the ith input variable without allowing to make the distinction between the two cases, non-linearity and interactions. As presented in Saltelli et al. (2008), the sensitivity is measured using:

$$\mu_i = \frac{1}{r} \sum_{k=1}^{r} d_i \qquad \sigma_i = \sqrt{\frac{1}{r} \sum_{k=1}^{r} (d_i - \mu)^2} \qquad \mu_i^* = \frac{1}{r} \sum_{k=1}^{r} |d_i|. \tag{5.3}$$

Before starting the screening strategy, an experiment is performed with the aim to find an appropriate number of trajectories. Since the two optimization tasks differ in the number of input variables, the OT approach is applied in both cases. We

compare the robustness of the sensitivity results and their variation while the size of the optimal set of trajectories is changing. We first examine the sensitivity of each variable by setting $r = 2$. To overcome the issue of the opposite signs, we use the mean value of the absolute EEs, which also indicates the importance of the input parameters (Campolongo et al., 2011). The results are shown in Fig. 5.3. As the size of the optimal set is increasing, the width of the error bars becomes narrower. Setting the size of the optimal set of trajectories any value greater than $r = 6$ for the bore height case and $r = 8$ for the run-up case, would result in performing unnecessary model runs as the accuracy of the results is not improved.

For completeness, the Morris screening strategy is applied in both cases. The input variables are ranked in order of importance and, according to the classification scheme proposed in Sanchez et al. (2014) which is based on the ratio σ/μ^*, are characterised in terms of linearity ($\sigma/\mu^* < 0.1$), monotony ($0.5 > \sigma/\mu^* > 0.1$) or possible interactions ($\sigma/\mu^* > 1$). The classification scheme is based on the assumption that the EEs are normally distributed and that 95% of EEs are within the range ($\mu^* \pm 1.96\sigma$). For example, if $\sigma/\mu^* < 0.1$ most EEs (95%) are in the range of $\pm 20\%$ around μ^* making them almost constant and this indicates that the input variable has almost linear effect on the model. Fig 5.4 shows the Morris screening results for both cases. By plotting three straight lines of slopes $\sigma/\mu^* = 0.1, 0.5$ and 1, the elementary effect scatter plot (right part of fig 5.4) is used to graphically identify the effect that each input variable has on the model (Sanchez et al., 2014).

For the bore height case, all the input variables appear with significant influence, with the most influential one being the the significant wave height (*Hs*). None of them show a strong linear effect or possible interactions with at least one other variable. A linear relationship between the water depth over the reef (*h2*) and the bore height can be stated due to the relatively low standard deviation. For the wave run-up case, the most influential input variables are the beach slope (*bet*) and the significant wave height (*Hs*). The fore-reef slope (*alp*), water depth over the reef

(*h2*) and peak period (*Tp*) are less important but still with a significant influence on the model's output. On the other hand, the water depth offshore (*h1*) can be classified as negligible, without any impact on BOSZ and therefore, can be fixed during the optimization procedure. Except of the beach slope and the significant wave height, where their behaviour can be characterized closer to linear, all the other input variables show a non-linear influence and/or interactions with other parameters ($\sigma/\mu^* > 0.5$).

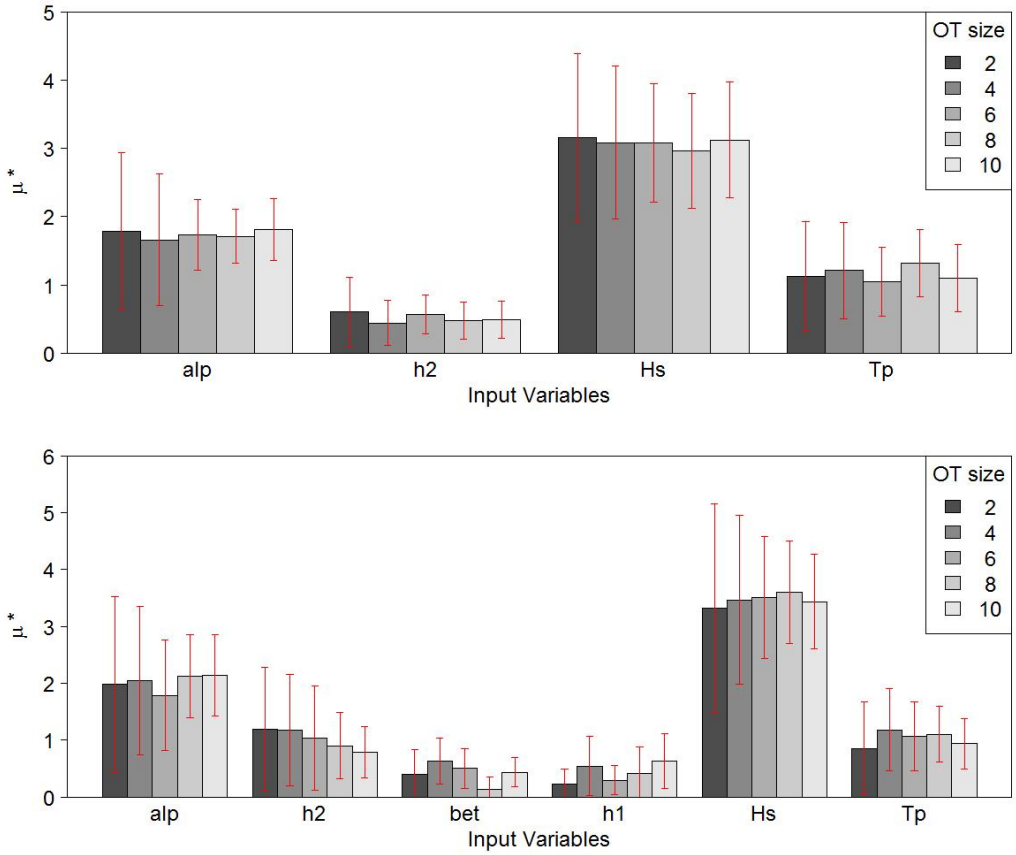


Figure 5.3: Robustness of the sensitivity results at different sizes of the optimal set of trajectories. Red lines show the error bars of the mean absolute elementary effects (μ^*). *Top panels*: for Bore height, *Bottom panels*: for Wave run-up.

5.3.2 Extreme bore height

After performing 200 simulations and using optim-MICE, the maximum bore height obtained is 7m (Fig. 5.6). To get a better understanding which of the input

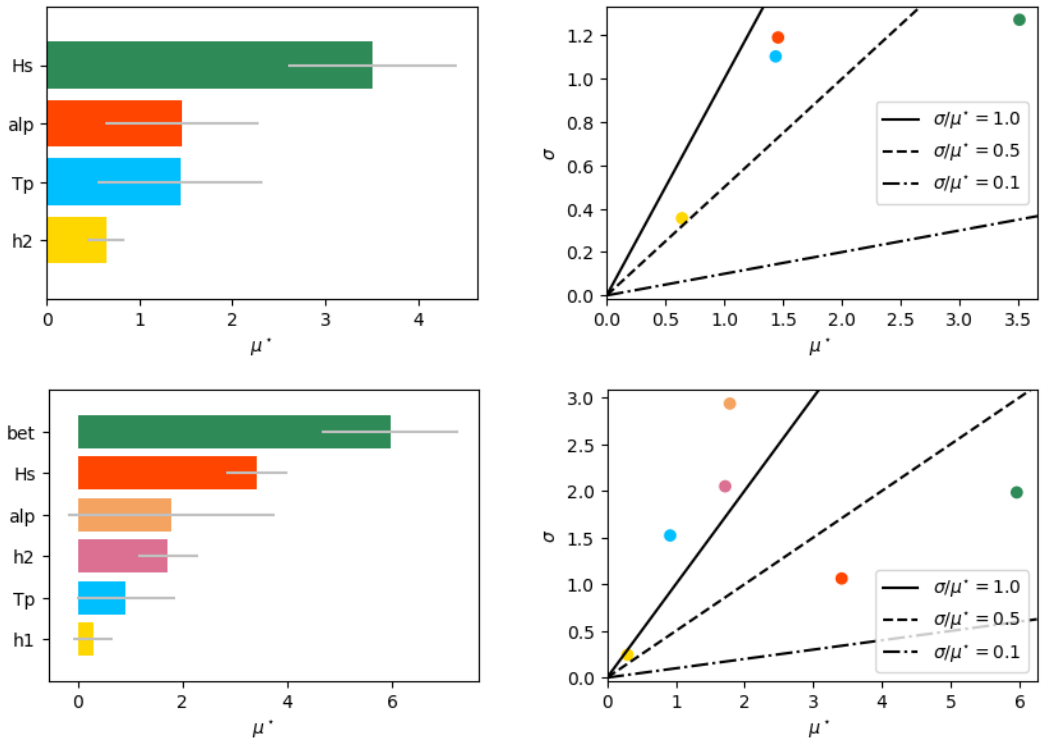


Figure 5.4: Results for the screening of input variables with the Morris method. Grey lines show the error bars of the mean absolute elementary effects (μ^*). *top*: for Bore height, *bottom*: for Wave run-up.

variables control the bore height and how, a local sensitivity analysis around the maximum bore height is performed using 500 runs of the last fitted Gaussian Process emulator. This allows us to carry out a near instantaneous sensitivity analysis (in less than 1 s), whereas running BOSZ around the maximum for 500 of inputs would be computationally costly, taking hours or days. The four panels in Fig. 5.5 shows the behavior of the maximum bore height over the reef flat as a function of the four input parameters reef slope (*alp*), water depth over the reef (*h2*), significant wave height (*Hs*) and peak period (*Tp*), under the condition that only one of the four parameters is changed, taking values across its range, and all others stay constant at the value where the maximum bore height is recorded.

It is not very surprising that the bore height increases with the significant wave height, i.e. the incident offshore wave energy. For a given set of peak period,

reef slope and reef depth, the increase in bore height is close to linear with the value of significant wave height. However, the other variables involved show a different trend. So does the effect of an increasing peak period tend to taper off beyond Tp $\sim$ 16 sec. The offshore wavelength increases quadratically with the peak period. As the peak period increases, the overall wave spectrum then contains significantly more energy from longer waves in comparison to a spectrum with a shorter peak period. At the same time, the overall composition of wave energy in the spectrum shifts towards longer wave groups. Such wave groups can be seen as reoccurring pulses (above mean water level) and lolls (below mean water level) of several waves due to the interaction of individual waves with each other. These wave groups are of much longer period than the individual swell waves and they are therefore called infra-gravity (IG) waves. IG waves can be of very different nature. A way to distinguish them is to classify them into two groups of bound and free IG waves. The IG waves resulting from the composition of the spectrum are inherently connected to the shape of the swell spectrum and they are therefore bound IG waves. In contrast, IG waves released after wave breaking are mostly free waves. In many cases, the presence of large wave groups also results in large breaking waves and run-up. In case of the maximum bore height it is very likely that the highest values are reached when the most energetic wave group is present. For the tested configurations, the increase in wave groups does not lead to a steady increase in observed bore height after wave breaking occurs over the reef flat.

A similar, but yet more drastic effect can be observed when only the water depth over the reef is changed. Very shallow water over the reef does not lead to the largest bore heights, mainly because the height of a hydraulic jump depends on the water level at both sides of the discontinuity and the height of the jump itself. Over nearly dry bed, waves move as sheet flow rather than as a pronounced bore. However, for our particular configuration, water depths larger than $\sim$ $1m$ have an adverse effect on the bore height. This is due to the fact that the incoming energy is distributed over a larger mass of water. Instead of having a jamming effect, the

energy is absorbed by a larger mass of water that reduces the height of the bore.

The influence of the reef slope on the bore height follows a different trend in comparison to the other parameters. Starting off with a very gentle slope (right side of panel 1 in Fig 5.5), the zone where waves break is relatively wide. The water depth is a controlling factor for the initiation of wave breaking, i.e. individual waves of various heights start breaking in different water depths. With a gentle bathymetry slope, this leads to a wide spread area where wave breaking is possible and, in turn, a more pronounced dissipation zone. As the slope increases, the horizontal area of wave breaking shrinks and most waves tend to break around the same position over the slope. At the same time, wave shoaling starts to become effective and increases the wave height before breaking. Shoaling is a group wave process, i.e. it acts on the underlying bound IG waves. With very steep slopes, parts of the wave energy is reflected and the bore height behind the breaking zone does not further increase.

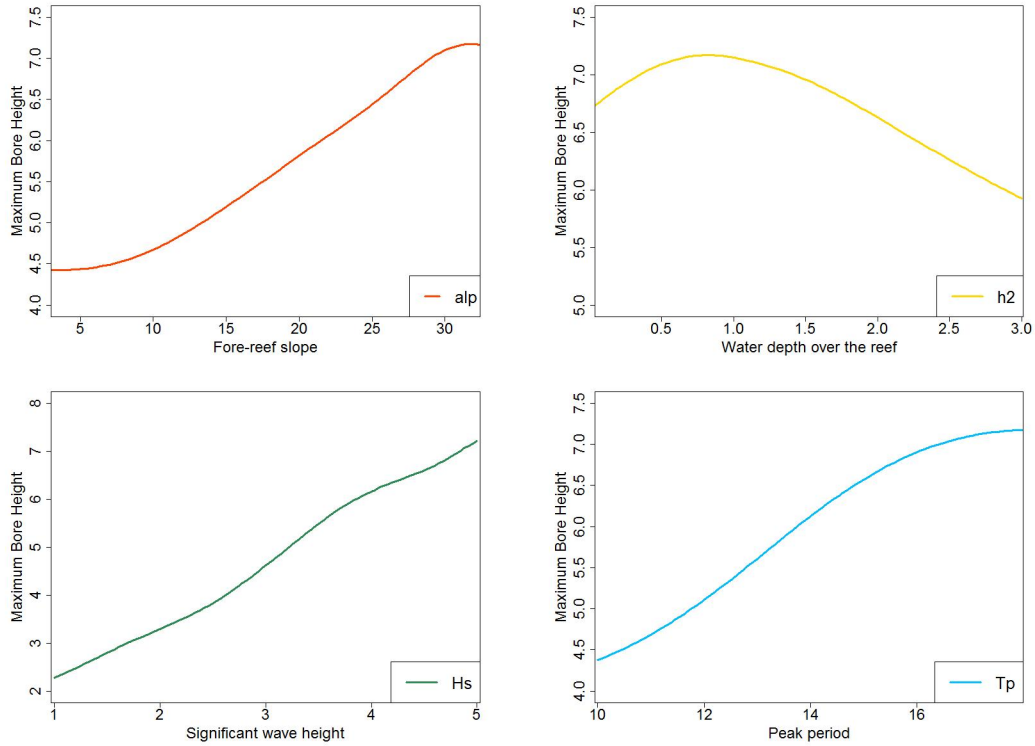


Figure 5.5: Local sensitivity of the maximum bore height on the fore-reef slope (alp), water depth over the reef ($h2$), significant wave height (Hs), peak period (Tp).

5.3.3 Extreme wave run-up

After performing 250 simulations and using optim-MICE, the maximum wave run-up obtained is 12m (Fig. 5.6). The behavior of the maximum run-up in dependence of the remaining parameters can be analyzed in a similar fashion as in the bore height case. By using the last fitted Gaussian Process emulator, a local sensitivity analysis around the maximum run-up is performed. The model was setup in an almost identical way compared to the bore height analysis - with the exception that a dry slope was added to the right side of the reef slope where waves can freely run up on after having propagated over the reef flat. The four panels in Fig. 5.7 show the behavior of the maximum run-up over the dry slope as a function of the other four input parameters significant wave height (Hs), peak period (Tp), water depth over the reef ($h2$), and bathymetry slopes, fore-reef slope (alp) and beach slope (bet), under the condition that only one of the four parameters is changed, taking values across its range, and all others remain constant at the value where the maximum run-up is recorded.

In contrast to the maximum bore height, the highest run-up does not exhibit a near-linear relationship with the incoming wave height (Hs). The maximum and minimum run-up are not found at the highest and lowest significant wave height of $5m$ and $1m$ respectively, but instead they occur at about 4.5m and 1.7m. In between the two values, a strong increase in run-up can be observed.

The dependence of the run-up on the peak period (Tp) and on the water depth over the reef ($h2$) follows a similar trend as what can be seen for the maximum bore height. Especially for the water depth of the reef, the highest run-up coincides approximately with the presence of the highest bore over the reef. This is somehow intuitive because there is no obstruction of the propagating bores on their way towards the run-up slope; i.e. an approaching large bore will most likely run up far on the slope unless an opposing flow from a previous drawdown stage reduces the flow momentum.

For the two slopes (fore-reef and run-up slope) we can conclude that a gentle slope generally leads to smaller run-up values than a steep slope. The reasons are different from what influences the maximum bore height. The lower run-up heights over gentle topographic slopes result mainly from the fact that wave breaking can take place over a longer duration and distance. Though, no bottom friction was implemented in the study, the effects of frictional loss are more pronounced over gentle slopes than over steep slopes. In other words, if frictional losses were accounted for, the dashed line in the upper left panel of Fig. 5.7 would be steeper.

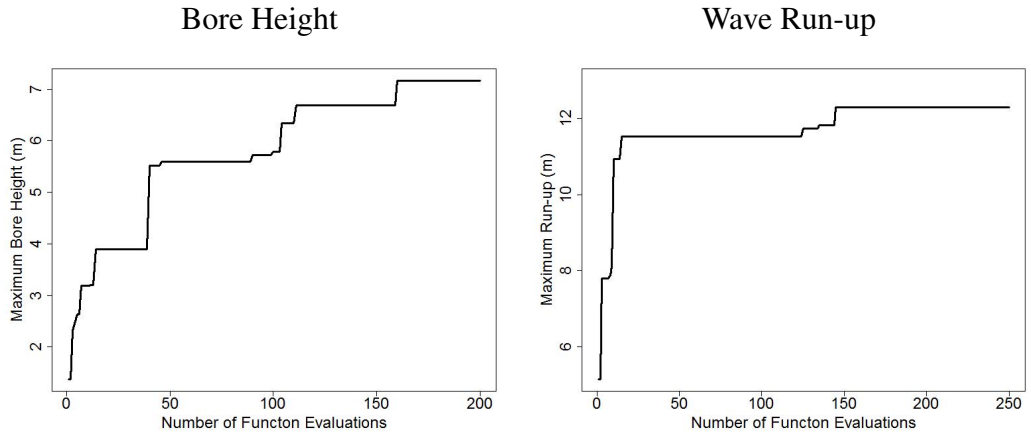


Figure 5.6: Optimization of BOSZ using the optim-MICE algorithm. *left*: Maximum Bore Height versus total function evaluations. *right*: Maximum Run-up versus total function evaluations.

5.3.4 Total effect of dependencies on run-up

Another way to look at the influences on the maximum wave run-up on a dry slope is the non-dimensional surf similarity parameter or Iribarren number expressed as $Ir = \tan alp/\sqrt{Hs/Lo}$, where $L_0 = 2Tp^2/2\pi$. Originally designed to give an estimate of wave breaker type and wave breaking intensity as a function of beach slope and offshore wave steepness. In general, low values indicate a gentle 'spilling' type of wave breaking, whereas high Iribarren numbers show evidence of plunging and even collapsing breakers - a type of wave breaking that occurs instantaneously

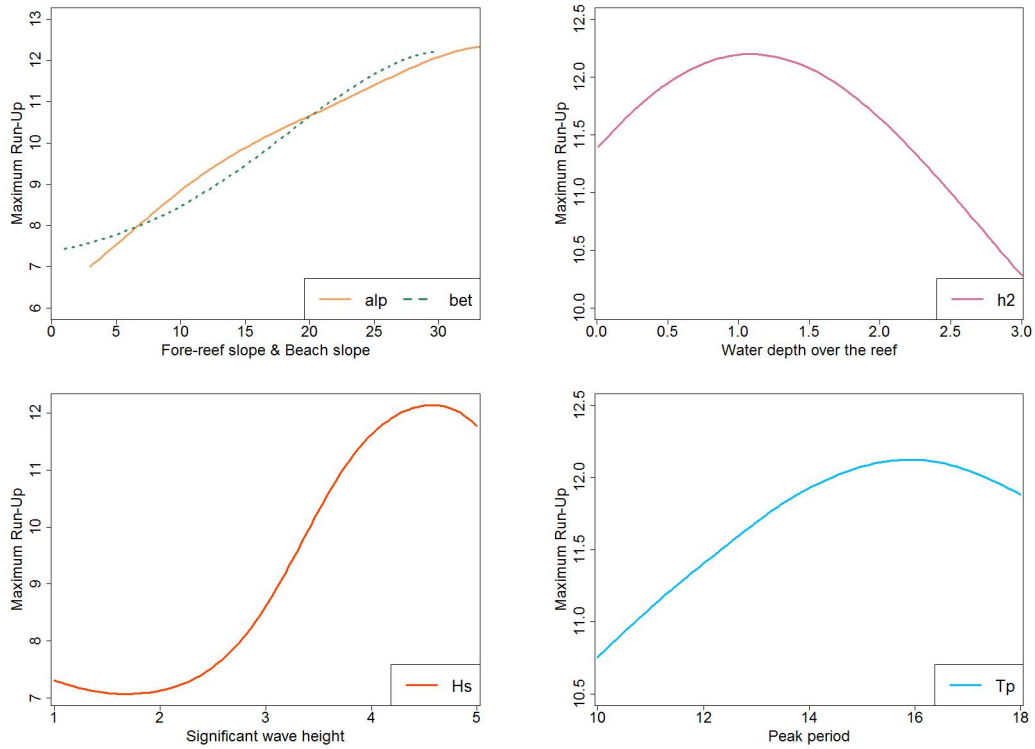


Figure 5.7: Local sensitivity of the maximum run-up on the fore-reef slope (alp), beach slope (bet), water depth over the reef ($h2$), significant wave height (Hs), peak period (Tp).

in contrast to the long-lasting spilling breakers. For a fixed slope, a lower wave steepness results in more abrupt wave breaking types. An extreme example is a tsunami wave, which is usually very long but of low amplitude. Once it approaches shallow water, it shoals over a long distance and ultimately breaks abruptly into a bore.

With respect to the presented case study, we generally observe high run-up values for low Iribarren numbers. Fig. 5.8(a) shows that the maximum run-up occurs for rather low ratios of reef depth to offshore wave height (significant wave height Hs is high in comparison to the water depth over the reef $h2$) in combination with low Iribarren numbers. A similar behavior can be observed in Fig. 5.8(b) where low Iribarren numbers lead to high run-up values if the ratio of bore height over the reef to offshore wave height is small. Both trends can be explained through

the mechanisms of energy dissipation in breaking waves. A low bore height to significant wave height indicates rather low dissipation rates of breaking waves in relation to the initial energy level. As the bores approach the run-up slope, they describe spilling rather than collapsing breakers and the energy dissipation is not instantaneous. The same is true for low ratios of water depth over the reef to offshore wave height (Fig. 5.8(a)). This leads to relatively high run-up heights.

Along the same lines, Fig. 5.8(c) shows that high run-up values can be found for low Iribarren numbers and low ratios of fore-reef slope to significant wave heights. In case of large incoming waves over gentle slopes, the waves approach rather as spilling breakers over a broad surf zone. It should be pointed out that the run-up values are distributed over a wide range of surf similarity values, i.e. low Iribarren numbers do not automatically lead to high run-up values. However, the largest run-up values fall into the category of low Iribarren numbers and no scenario in this case study shows very high run-up heights in combination with high surf similarity parameters.

5.4 Summary

Employing optim-MICE for the current optimization task gave us the opportunity to find the maximum breaking wave (bore) height and the maximum run-up in less computational time. In two idealised settings, we efficiently identify the conditions that create the largest storm waves at the coast using a minimal number of simulations.

This case study uses an idealized bathymetry whose main features characterize a fringing reef. The computation of a suite of typical storm waves over this bathymetry shows that maximum breaking wave heights and runup on a straight beach behind the reef follow particular trends. Most importantly, the run-up on the beach after the wave breaking process is not linearly related to the incoming wave

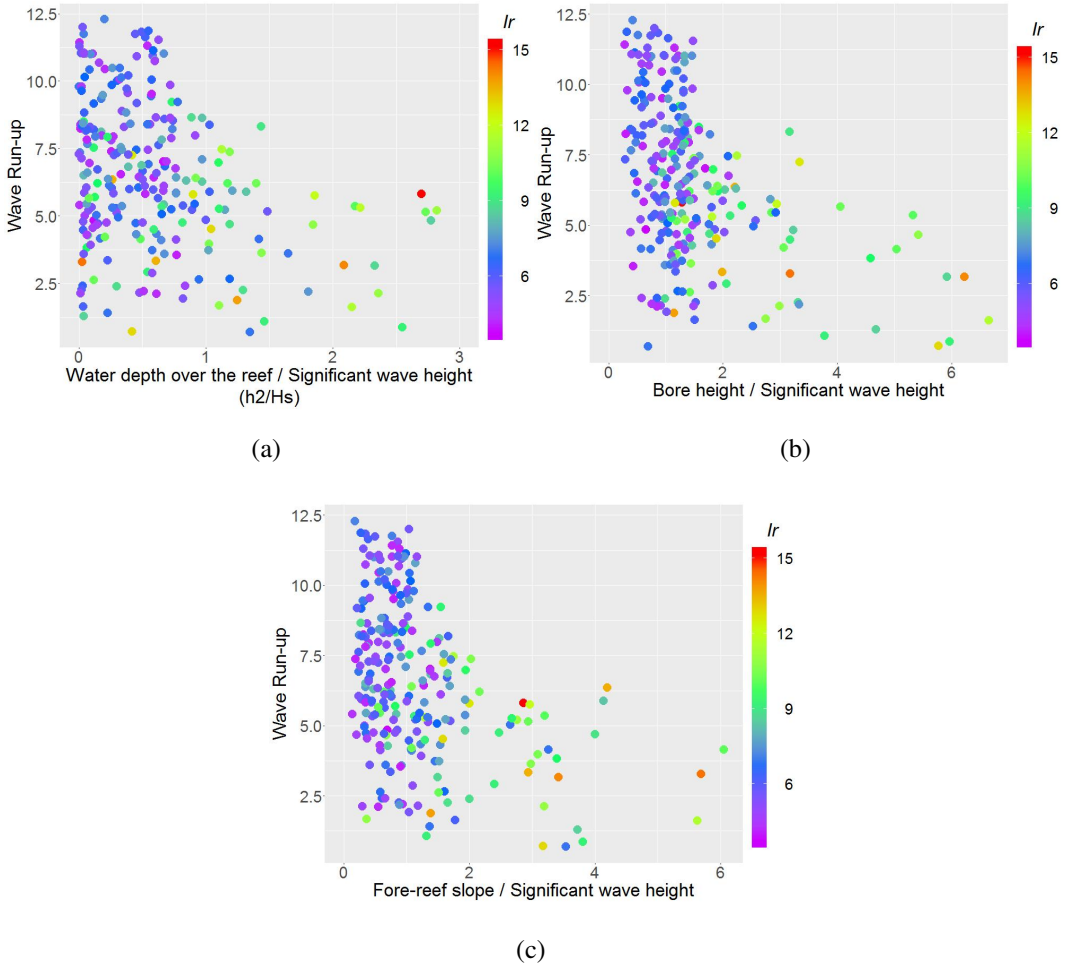


Figure 5.8: Wave run-up as a function of (a) water depth over the reef to significant wave height, (b) bore height to significant wave height and (c) fore reef slope to significant wave height. The bar indicates the Iribarren number *Ir* (the surf similarity parameter) calculated with the fore-reef slope, significant wave height and wavelength.

energy, whereas the breaking wave height is close to linearly related to the height of the approaching waves. The water depth over the reef is a strong controlling parameter for both breaking wave height and run-up. Counter-intuitively, the highest values do not occur at the highest water level, but in fact at relatively low levels (around 1m in our case study). This can be explained with the generation of infragravity waves through wave breaking and their effect on the nearshore dynamics, which are highly nonlinear processes.

Finally, with the presented optimization algorithm, it is possible to shorten the

computation time significantly compared to what is usually required to run through a permutation of scenarios in order to find a worst case scenario. This approach will greatly help assess quickly the potential of coastal hazards and thus improve future hazard mitigation efforts.

Chapter 6

Conclusion

6.1 Summary

The current thesis has been mainly motivated by the challenge of optimizing a black-box function where traditional mathematical approaches cannot be taken, the knowledge about the objective function is limited, and often does not exist, and each function evaluation is computationally expensive. To tackle this problem, this study proposed optim-MICE, a novel surrogate-based optimization scheme. Taking advantage of parallelism of the evaluation of the unknown function, the uncertain regions are explored simultaneously, and a batch of input points is chosen using Mutual Information for Computer Experiments (MICE), a sequential design algorithm which maximizes the information theoretic Mutual Information over the input space. Specifically, optim-MICE explores the entire input space and focuses on the region where the maximum belongs in with high probability. The objective function is only evaluated at a batch of input points that are chosen to yield the maximum information about it.

Overall, incorporating the MICE criterion in the GP framework seems to be more computationally beneficial compared to the state-of-the-art heuristics. The optim-MICE method increases the confidence in getting a solution close to the true optimum in fewer function evaluations. Regardless of the dimensionality and com-

plexity of the computer model, with optim-MICE the search process for identifying the uncertain region is more efficient and the regret decays faster compared to the alternatives. The total computational time needed to find the true optimum and achieve convergence of the regret is less than the other approaches.

The performance of the proposed optimization scheme is affected by the algorithm settings. A complex and a high-dimensional computer model clearly needs more iterations and a bigger batch size than a simple and low-dimensional function. To get the most of the proposed optimization scheme, a balance between the number of iterations and the batch size is needed according to the complexity that a function might have. If one performs a lot of iterations without exploring enough, the information gain about the unknown function at each time step is limited as the number of input points added in the design is small and the objective function is evaluated only a few times. Furthermore, the advantage of MICE is not fully utilized and a certain computational cost is added, without necessarily needed, such as the re-estimation of the hyper-parameters of the surrogate model.

When making use of a large batch size, the uncertain region is definitely explored more and the chance to find the true optimum in fewer function evaluations is higher. But, choosing a larger batch size than what is needed, an amount of computational time is wasted as the algorithm is forced to stay in a region which might not be of interest anymore - and has already been discovered from the first exploration steps - or in a region which has already been explored and any additional information will not add value. In both cases, it is unavoidable that a number of function evaluations are performed without obtaining any progress.

To keep the number of function evaluations as low as possible when a large batch size is chosen, based on the two experiments performed in Chapter 4, it is worth scaling the objective function vertically by a small factor (e.g. 0.5). This can only be generalised if more experiments are performed examining different functions

with different shapes and physical properties. Based on the two examined cases, by shrinking the uncertain region in the vertical direction, the information obtained by each new point added in the design at each iteration seems to be more valuable as it is more likely to be closer to the true optimum. In terms of *Nsearch* and *Ncand*, things are more straightforward. Regardless of dimensionality and complexity of the function, having a large number of input points spread around the search space and choosing to examine a big set of candidate points with the MICE criterion could lead us to more accurate results without wasting more computational resources.

The first surrogate-based optimization of storm waves was presented in the current thesis. The proposed algorithm was used to identify the extrema of coastal storm waves over a reef-type bathymetry, for two quantities: breaking wave height and local wave run-up. The hypothetical setting was similar to what governed the wave event that destroyed the town of Hernani during Typhoon Haiyan. The capability of the computer model in handling various wave processes, gave us the opportunity to explore different aspects of storm wave extrema, such as the maximum bore height and the maximum run-up, at a local level. Due to the computational complexity of the computer model, finding the combination of the controlling parameters that leads to the worst-case scenario would require a large number of individuals runs. However, by employing optim-MICE, the computation time shorten significantly.

The computational efficiency of the optim-MICE algorithm helped us efficiently identify and explore the important regions fast and obtain the maximum run-up and bore height in the lowest possible number of function evaluations (model runs). Overall, the computational analysis performed in Chapter 3 showed that the total number of individual runs needed to find the maximum, or a solution close to the maximum, is increased by 20% for GP-UCB-PE, and 75% for qEGO, when compared to optim-MICE. Considering now only the cases with higher dimensions, on average, getting a solution close to the true optimum with GP-UCB-PE required 50 more function evaluations and 140 with qEGO, compared to optim-MICE (with

a 250 run budget). Translating this into computational time, and knowing that the current study needed about 10 hours on a cluster of 4 cores, such an optimization task would be expected to be completed in approximately 12 and 16 hours if GP-UCB-PE and qEGO would have been used, respectively.

The more complex the computer set-up, e.g. for realistic coastal hazard assessments with one run taking hours not minutes, the more computational resources are required to complete the overall optimization task and therefore, it is crucial to ensure that a good solution can be achieved in the lowest possible computational time. Considering the overall performance of the proposed optimization scheme, which was demonstrated in the current study, optim-MICE is better at imparting confidence that the maximum, or a solution close to the maximum, can be achieved.

6.2 Future work

In the current work, the proposed optimization algorithm was compared with state-of-the-art heuristics and to keep consistency all the algorithm settings were chosen to be the same. One of the important settings is the size of the set of candidate points. Due to the limited computational budget, optim-MICE, compared to the alternative optimization schemes, could only examine a small number of candidate points. Despite that, it still outperforms the others and its efficiency was proved in various computational experiments. It is true that having a large number of candidate points increases the chances to find the true optimum, or a solution close to the true optimum, with the lowest number of function evaluations. But unavoidably, the computational time increases as all the candidate points have to be examined with the MICE criterion. Also, a lot of GP predictions need to be performed until the input point that provides the maximum increase in mutual information is obtained. This step is actually the most computational heavy part of optim-MICE. To further speed up the entire proposed optimization algorithm, an interesting extension would be to make use of GPU (Graphics Processing Unit) acceleration. GPU is a type

of specialised computation hardware consisting of many small processing units (cores) able to handle multiple tasks in parallel efficiently. Its computational power will give us the opportunity not only to perform exact GP inference for large designs but also to compute predictive means and variances at thousands new points. By exploiting the computational capabilities of the GPU, we can perform fast GP predictions and make full use of the efficiency of optim-MICE. By examining more candidate points with the MICE criterion at each time-step of the algorithm, the true optimum can be found in even less function evaluations minimizing the total computational resources needed.

Another extension could be to provide theoretical guarantees through the upper bounds for cumulative regret of optim-MICE, which can be seen as the convergence rates of the GP optimization. The technical connection between the bandit setting and experimental design has been shown in various established algorithms where the theoretical bounds of their cumulative regret have been provided. Since the proposed optimization algorithm chooses a batch of input points based on the MICE criterion, a full information theoretic mutual information measure, a new regret bound needs to be calculated. In addition, a full theoretical framework under the bandit setting and the mutual information measure can also be provided for GP optimization with commonly-used covariance functions, such as Squared Exponential and Matérn covariance function.

In an optimization problem, such as the computational experiments presented in the current study, we are interested in finding the true optimum or otherwise, the feasible solution: the solution that optimizes the objective function. What if a problem involves the simultaneous optimization of several objective functions where these functions are evaluated by expensive deterministic computer models? An example could be the study for coastal wave heights and currents across an entire region, not one output type at one location at a time as in Chapter 5. Therefore, a possible future study is the extension of optim-MICE into the multi-objective setting. A unique

solution that optimizes all the objective functions simultaneously may not exist, instead a possibly infinite number of optimal solutions exists. In a multi-objective setting, also known as Pareto optimization, we are interested in finding the Pareto optimal. A solution is called Pareto optimal if none of the objective functions can be improved in value without degrading some of the other objective values. Given the computational efficiency of integrating the MICE criterion into a surrogate-based optimization scheme, this extension could possibly reduce significantly the computational burden of optimizing multiple objective functions.

Bibliography

Abramson, M. A. & Audet, C. (2006). Convergence of mesh adaptive direct search to second-order stationary points. *SIAM Journal on Optimization*, 17(2), 606–619.

Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for learning. *Machine Learning*, 50(1), 5–43.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov), 397–422.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2-3), 235–256.

Aye, S. A. & Heyns, P. S. (2018). Prognostics of slow speed bearings using a composite integrated gaussian process regression model. *International Journal of Production Research*, 56(14), 4860–4873.

Azimi, J., Fern, A., & Fern, X. Z. (2010). Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems* (pp. 109–117).

Azimi, J., Jalali, A., & Zhang-fern, X. (2012). Hybrid batch Bayesian optimization. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)* (pp. 1215–1222).

Ba, S. & Joseph, V. R. (2011). Multi-layer designs for computer experiments. *Journal of the American Statistical Association*, 106(495), 1139–1149.

Bachoc, F. (2013). Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification. *Computational Statistics & Data Analysis*, 66, 55–69.

Bastos, L. S. & O'Hagan, A. (2009). Diagnostics for Gaussian process emulators. *Technometrics*, 51(4), 425–438.

Beck, J. & Guillas, S. (2016). Sequential design with mutual information for computer experiments (MICE): emulation of a tsunami model. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1), 739–766.

Behrens, J. & Dias, F. (2015). New computational methods in tsunami science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 373(2053), 20140382.

Bengtsson, L., Hodges, K. I., & Keenlyside, N. (2009). Will extratropical storms intensify in a warmer climate? *Journal of Climate*, 22(9), 2276–2301.

Bilionis, I., Zabaras, N., Konomi, B. A., & Lin, G. (2013). Multi-output separable Gaussian process: towards an efficient, fully bayesian paradigm for uncertainty quantification. *Journal of Computational Physics*, 241, 212–239.

Boukouvala, F., Misener, R., & Floudas, C. A. (2016). Global optimization advances in mixed-integer nonlinear programming, minlp, and constrained derivative-free optimization, cdfo. *European Journal of Operational Research*, 252(3), 701–727.

Boukouvalas, A., Gosling, J. P., & Maruri-Aguilar, H. (2014). An efficient screening method for computer experiments. *Technometrics*, 56(4), 422–431.

Bowman, V. E. & Woods, D. C. (2016). Emulation of multivariate simulators using thin-plate splines with application to atmospheric dispersion. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1), 1323–1344.

Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Bubeck, S., Cesa-Bianchi, N., et al. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1), 1–122.

Bubeck, S., Munos, R., Stoltz, G., & Szepesvári, C. (2011). X-armed bandits. *Journal of Machine Learning Research*, 12(May), 1655–1695.

Bull, A. D. (2011). Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(Oct), 2879–2904.

Campolongo, F., Cariboni, J., & Saltelli, A. (2007). An effective screening design for sensitivity analysis of large models. *Environmental modelling & software*, 22(10), 1509–1518.

Campolongo, F., Saltelli, A., & Cariboni, J. (2011). From screening to quantitative sensitivity analysis. a unified approach. *Computer Physics Communications*, 182(4), 978–988.

Caselton, W. F. & Zidek, J. V. (1984). Optimal monitoring network designs. *Statistics & Probability Letters*, 2(4), 223–227.

Chaloner, K. & Verdinelli, I. (1995). Bayesian experimental design: A review. *Statistical Science*, (pp. 273–304).

Chen, J., Xin, B., Peng, Z., Dou, L., & Zhang, J. (2009). Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(3), 680–691.

Chevalier, C. & Ginsbourger, D. (2013). Fast computation of the multi-points expected improvement with applications in batch selection. In *International Conference on Learning and Intelligent Optimization* (pp. 59–69).: Springer.

Ciaurri, D. E., Isebor, O., & Durlofsky, L. (2010). Application of derivative-free methodologies to generally constrained oil production optimization problems. *Procedia Computer Science*, 1(1), 1301 – 1310.

Cohn, D. (1996). Neural network exploration using optimal experiment design. *Neural networks: the official journal of the International Neural Network Society*, 9(6), 1071.

Conn, A., Scheinberg, K., & Vicente, L. (2009a). *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics.

Conn, A. R., Scheinberg, K., & Vicente, L. N. (2009b). Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points. *SIAM Journal on Optimization*, 20(1), 387–415.

Contal, E., Buffoni, D., Robicquet, A., & Vayatis, N. (2013). Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 225–240).: Springer.

Contal, E. & Vayatis, N. (2016). Stochastic process bandits: Upper confidence bounds algorithms via generic chaining. *arXiv preprint arXiv:1602.04976*.

Conti, S. & O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models. *Journal of statistical planning and inference*, 140(3), 640–651.

Cover, T. M. & Thomas, J. A. (2006). *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience.

Cox, D. D. & John, S. (1992). A statistical method for global optimization. In *Systems, Man and Cybernetics, 1992., IEEE International Conference on* (pp. 1241–1246).: IEEE.

Črepinšek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3), 35.

Currin, C., Mitchell, T., Morris, M., & Ylvisaker, D. (1988). *A Bayesian Approach to the Design and Analysis of Computer Experiments*. Technical report, ORNL Oak Ridge National Laboratory (US).

Currin, C., Mitchell, T., Morris, M., & Ylvisaker, D. (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416), 953–963.

Dani, V., Hayes, T. P., & Kakade, S. M. (2008a). Stochastic linear optimization under bandit feedback. In *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, July 9-12, 2008* (pp. 355–366).

Dani, V., Kakade, S. M., & Hayes, T. P. (2008b). The price of bandit information for online optimization. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in Neural Information Processing Systems 20* (pp. 345–352). Curran Associates, Inc.

Daxberger, E. A. & Low, B. K. H. (2017). Distributed batch Gaussian process optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 951–960).: JMLR. org.

De Waal, D. & Van Gelder, P. (2005). Modelling of extreme wave heights and periods through copulas. *Extremes*, 8(4), 345–356.

Desautels, T., Krause, A., & Burdick, J. W. (2014). Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(1), 3873–3923.

Ellis, J. T. & Sherman, D. J. (2015). Perspectives on coastal and marine hazards and disasters. In *Coastal and Marine Hazards, Risks, and Disasters* (pp. 1–13). Elsevier.

Feng, J., Li, D., Li, Y., Liu, Q., & Wang, A. (2018). Storm surge variation along the coast of the Bohai Sea. *Scientific reports*, 8(1), 11309.

Ferrario, F., Beck, M. W., Storlazzi, C. D., Micheli, F., Shepard, C. C., & Airoldi, L. (2014). The effectiveness of coral reefs for coastal hazard risk reduction and adaptation. *Nature communications*, 5, 3794.

Forrester, A., Sobester, A., & Keane, A. (2008). *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons.

Frazier, P. I. (2018). A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*, (pp. 1–67).

Garbuno-Inigo, A., DiazDelaO, F., & Zuev, K. (2016). Gaussian process hyperparameter estimation using parallel asymptotically independent markov sampling. *Computational Statistics & Data Analysis*, 103, 367–383.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., & Rubin, D. (2013). *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.

Genton, M. G. (2001). Classes of kernels for machine learning: a statistics perspective. *Journal of machine learning research*, 2(Dec), 299–312.

Gilks, W., Richardson, S., & Spiegelhalter, D. (1995). *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC Interdisciplinary Statistics. Taylor & Francis.

Ginsbourger, D., Dupuy, D., Badea, A., Carraro, L., & Roustant, O. (2009). A note on the choice and the estimation of kriging models for the analysis of deterministic computer experiments. *Applied Stochastic Models in Business and Industry*, 25(2), 115–131.

Ginsbourger, D., Le Riche, R., & Carraro, L. (2008). A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. *HAL*.

Giunta, A. A., Balabanov, V., Haim, D., Grossman, B., Mason, W. H., Watson, L. T., & Haftka, R. T. (1997). Multidisciplinary optimisation of a supersonic transport using design of experiments theory and response surface modelling. *The Aeronautical Journal (1968)*, 101(1008), 347–356.

Gramacy, R. B. (2007). tgp: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models. *Journal of Statistical Software*, 19(i09).

Gramacy, R. B. & Lee, H. K. (2009). Adaptive design and analysis of supercomputer experiments. *Technometrics*, 51(2), 130–145.

Gramacy, R. B. & Lian, H. (2012). Gaussian process single-index models as emulators for computer experiments. *Technometrics*, 54(1), 30–41.

Grünewälder, S., Audibert, J.-y., Opper, M., & Shawe-Taylor, J. (2010). Regret bounds for Gaussian process bandit problems. In *International Conference on Artificial Intelligence and Statistics* (pp. 273–280).

Guillas, S., Sarri, A., Day, S. J., Liu, X., Dias, F., et al. (2018). Functional emulation of high resolution tsunami modelling over cascadia. *The Annals of Applied Statistics*, 12(4), 2023–2053.

Gutmann, H.-M. (2001). A radial basis function method for global optimization. *Journal of global optimization*, 19(3), 201–227.

Hakkou, M., Maanan, M., Belrhaba, T., Leone, F., Benmohammadi, A., Zourarah, B., et al. (2019). Assess and mapping the flooding hazards using geospatial tools and empirical model along Kenitra Coast, Morocco. *Ocean & coastal management*, 169, 264–272.

Handcock, M. S. & Stein, M. L. (1993). A Bayesian analysis of kriging. *Technometrics*, 35(4), 403–410.

Hasselmann, K., Barnett, T., Bouws, E., Carlson, H., Cartwright, D., Enke, K., Ewing, J., Gienapp, H., Hasselmann, D., Kruseman, P., et al. (1973). Measurements

of wind-wave growth and swell decay during the joint north sea wave project (JONSWAP). *Ergänzungsheft 8-12*.

Hoffman, M. D., Brochu, E., & de Freitas, N. (2011). Portfolio allocation for Bayesian optimization. In *UAI*.

Huang, D., Allen, T. T., Notz, W. I., & Zeng, N. (2006). Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of global optimization*, 34(3), 441–466.

Iooss, B. & Marrel, A. (2017). An efficient methodology for the analysis and modeling of computer experiments with large number of inputs. *arXiv preprint arXiv:1704.07090*.

Ishii, S., Yoshida, W., & Yoshimoto, J. (2002). Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural networks*, 15(4), 665–687.

Jamil, M. & Yang, X.-S. (2013). A literature survey of benchmark functions for global optimization problems. *arXiv preprint arXiv:1308.4008*.

Janusevskis, J., Le Riche, R., Ginsbourger, D., & Girdziusas, R. (2012). Expected improvements for the asynchronous parallel global optimization of expensive functions: Potentials and challenges. In *International Conference on Learning and Intelligent Optimization* (pp. 413–418).: Springer.

Ji, C., Zhang, Q., & Wu, Y. (2018). An empirical formula for maximum wave setup based on a coupled wave-current model. *Ocean Engineering*, 147, 215–226.

Johnson, M. E., Moore, L. M., & Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26(2), 131–148.

Jonathan, P., Ewans, K., & Randell, D. (2014). Non-stationary conditional extremes of northern north sea storm characteristics. *Environmetrics*, 25(3), 172–188.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4), 345–383.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4), 455–492.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4, 237–285.

Kandasamy, K., Dasarathy, G., Oliva, J. B., Schneider, J., & Póczos, B. (2016). Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems* (pp. 992–1000).

Kandasamy, K., Schneider, J., & Póczos, B. (2015). High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning* (pp. 295–304).

Kleinberg, R., Slivkins, A., & Upfal, E. (2008). Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing* (pp. 681–690).: ACM.

Kolda, T. G., Lewis, R. M., & Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3), 385–482.

Koziel, S., Ciaurri, D. E., & Leifsson, L. (2011). Surrogate-based methods. In *Computational Optimization, Methods and Algorithms* (pp. 33–59). Springer.

Krause, A. & Guestrin, C. E. (2012). Near-optimal nonmyopic value of information in graphical models. *arXiv preprint arXiv:1207.1394*.

Krause, A. & Ong, C. S. (2011). Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems* (pp. 2447–2455).

Krause, A., Singh, A., & Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb), 235–284.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1), 97–106.

Lai, T. L. & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1), 4–22.

Lam, C. Q. (2008). *Sequential adaptive designs in computer experiments for response surface model fit*. PhD thesis, The Ohio State University.

Lee, P. (2004). *Bayesian Statistics: An Introduction*. John Wiley & Sons.

Leijala, U., Björkqvist, J.-V., Johansson, M. M., Pellikka, H., Laakso, L., & Kahma, K. K. (2018). Combining probability distributions of sea level variations and wave run-up to evaluate coastal flooding risks. *Natural Hazards and Earth System Sciences*, 18(10), 2785–2799.

Li, F., Van Gelder, P., Ranasinghe, R., Callaghan, D., & Jongejan, R. (2014). Probabilistic modelling of extreme storms along the dutch coast. *Coastal Engineering*, 86, 1–13.

Lin, N., Emanuel, K., Oppenheimer, M., & Vanmarcke, E. (2012). Physically based assessment of hurricane surge threat under climate change. *Nature Climate Change*, 2(6), 462.

Lizotte, D. J. (2008). *Practical Bayesian optimization*. University of Alberta.

Lucidi, S. & Sciandrone, M. (2002). On the global convergence of derivative-free methods for unconstrained optimization. *SIAM Journal on Optimization*, 13(1), 97–116.

MacKay, D. J. (1992). Information-based objective functions for active data selection. *Neural computation*, 4(4), 590–604.

Mardia, K. & Watkins, A. (1989). On multimodality of the likelihood in the spatial linear model. *Biometrika*, 76(2), 289–295.

Mardia, K. V. & Marshall, R. J. (1984). Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika*, 71(1), 135–146.

Marin, O. (2005). *Designing computer experiments to estimate integrated response functions*. PhD thesis, The Ohio State University.

Marshall, L., Johnson, J. S., Mann, G. W., Lee, L., Dhomse, S. S., Regayre, L., Yoshioka, M., Carslaw, K. S., & Schmidt, A. (2018). Exploring how eruption source parameters affect volcanic radiative forcing using statistical emulation. *Journal of Geophysical Research: Atmospheres*.

May, S., Engel, M., Brill, D., Cuadra, C., Lagmay, A., Santiago, J., Suarez, J., Reyes, M., & Brückner, H. (2015). Block and boulder transport in Eastern Samar (Philippines) during supertyphoon Haiyan. *Earth Surface Dynamics*, 3(3).

McInnes, K., Walsh, K., Hubbert, G., & Beer, T. (2003). Impact of sea-level rise and storm surges on a coastal community. *Natural Hazards*, 30(2), 187–207.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2), 239–245.

Minasny, B. & McBratney, A. B. (2005). The matérn function as a general model for soil variograms. *Geoderma*, 128(3), 192–207.

Mockus, J., Tiesis, V., & Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129), 2.

Montoya, L. & Lynett, P. (2018). Tsunami versus infragravity surge: Comparison of the physical character of extreme runup. *Geophysical Research Letters*, 45(23), 12–982.

Mori, N., Kato, M., Kim, S., Mase, H., Shibutani, Y., Takemi, T., Tsuboki, K., & Yasuda, T. (2014). Local amplification of storm surge by super typhoon Haiyan in the Leyte Gulf. *Geophysical research letters*, 41(14), 5106–5113.

Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2), 161–174.

Muis, S., Verlaan, M., Winsemius, H. C., Aerts, J. C., & Ward, P. J. (2016). A global reanalysis of storm surges and extreme sea levels. *Nature communications*, 7, 11969.

Nelder, J. A. & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4), 308.

Ning, Y., Liu, W., Zhao, X., Zhang, Y., & Sun, Z. (2019). Study of irregular wave run-up over fringing reefs based on a shock-capturing boussinesq model. *Applied Ocean Research*, 84, 216–224.

Northrop, P. J., Attalides, N., & Jonathan, P. (2017). Cross-validatory extreme value threshold selection and uncertainty with application to ocean storm severity. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 66(1), 93–120.

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

Oeuvray, R. & Bierlaire, M. (2007). A new derivative-free algorithm for the medical image registration problem. *Int. J. Model. Simul.*, 27(2), 115–124.

O'Hagan, A. (2006). Bayesian analysis of computer code outputs: a tutorial. *Reliability Engineering & System Safety*, 91(10), 1290–1300.

Overstall, A. M. & Woods, D. C. (2016). Multivariate emulation of computer simulators: model selection and diagnostics with application to a humanitarian relief model. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 65(4), 483–505.

Park, H. & Cox, D. T. (2016). Empirical wave run-up formula for wave, storm surge and berm width. *Coastal Engineering*, 115, 67–78.

Parker, K., Ruggiero, P., Serafin, K. A., & Hill, D. F. (2019). Emulation as an approach for rapid estuarine modeling. *Coastal Engineering*, 150, 79–93.

Pronzato, L. & Müller, W. G. (2012). Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3), 681–701.

Qin, C., Klabjan, D., & Russo, D. (2017). Improving the expected improvement algorithm. In *Advances in Neural Information Processing Systems* (pp. 5381–5391).

Quataert, E., Storlazzi, C., Rooijen, A., Cheriton, O., & Dongeren, A. (2015). The influence of coral reefs and climate change on wave-driven flooding of tropical coastlines. *Geophysical Research Letters*, 42(15), 6407–6415.

Quirante, N., Javaloyes, J., & Caballero, J. A. (2015). Rigorous design of distillation columns using surrogate models based on kriging interpolation. *AIChE Journal*, 61(7), 2169–2187.

Rasmussen, C. E. & Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Ribera, P., García-Herrera, R., & Gimeno, L. (2008). Historical deadly typhoons in the philippines. *Weather*, 63(7), 194–199.

Rios, L. M. & Sahinidis, N. V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3), 1247–1293.

Robbins, H. (1985). Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers* (pp. 169–177). Springer.

Roeber, V. & Bricker, J. D. (2015). Destructive tsunami-like wave generated by surf beat over a coral reef during typhoon Haiyan. *Nature communications*, 6.

Roeber, V. & Cheung, K. F. (2012). Boussinesq-type model for energetic breaking waves in fringing reef environments. *Coastal Engineering*, 70, 1–20.

Romero, R. & Emanuel, K. (2017). Climate change and hurricane-like extratropical cyclones: Projections for north atlantic polar lows and medicanes based on cmip5 models. *Journal of Climate*, 30(1), 279–299.

Rougier, J., Guillas, S., Maute, A., & Richmond, A. D. (2009). Expert knowledge and multivariate emulation: The thermosphere–ionosphere electrodynamics general circulation model (TIE-GCM). *Technometrics*, 51(4), 414–424.

Roustant, O., Ginsbourger, D., & Deville, Y. (2012). Dicekriging, diceoptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of statistical software*, 51(1), 1–55.

Rueda, A., Camus, P., Tomás, A., Vitousek, S., & Méndez, F. (2016). A multivariate extreme wave and storm surge climate emulator based on weather patterns. *Ocean Modelling*, 104, 242–251.

Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989). Design and analysis of computer experiments. *Statist. Sci.*, 4(4), 409–423.

Salmanidou, D., Guillas, S., Georgiopoulou, A., & Dias, F. (2017). Statistical emulation of landslide-induced tsunamis at the rockall bank, NE atlantic. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2200), 20170026.

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., & Tarantola, S. (2008). *Global sensitivity analysis: the primer*. John Wiley & Sons.

Sammarco, P. & Renzi, E. (2008). Landslide tsunamis propagating along a plane beach. *Journal of Fluid Mechanics*, 598, 107–119.

Sanchez, D. G., Lacarrière, B., Musy, M., & Bourges, B. (2014). Application of sensitivity analysis in building energy simulations: Combining first-and second-order elementary effects methods. *Energy and Buildings*, 68, 741–750.

Santner, T. J., Williams, B. J., & Notz, W. (2003). *The Design and Analysis of Computer Experiments*. Springer-Verlag.

Sarkar, D., Contal, E., Vayatis, N., & Dias, F. (2016). Prediction and optimization of wave energy converter arrays using a machine learning approach. *Renewable Energy*, 97, 504–517.

Sarri, A., Guillas, S., & Dias, F. (2012). Statistical emulation of a tsunami model for sensitivity analysis and uncertainty quantification. *Natural Hazards and Earth System Sciences*, 12, 2003–2018.

Sartini, L., Besio, G., & Cassola, F. (2017). Spatio-temporal modelling of extreme wave heights in the mediterranean sea. *Ocean Modelling*, 117, 52–69.

Seo, S., Wallat, M., Graepel, T., & Obermayer, K. (2000). Gaussian process regression: Active data selection and test point rejection. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3 (pp. 241–246).

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1), 148–175.

Shimozono, T., Tajima, Y., Kennedy, A. B., Nobuoka, H., Sasaki, J., & Sato, S. (2015). Combined infragravity wave and sea-swell runup over fringing reefs by super typhoon Haiyan. *Journal of Geophysical Research: Oceans*, 120(6), 4463–4486.

Simpson, T., Lin, D., & Chen, W. (2001). Sampling strategies for computer experiments: design and analysis. *International Journal of Reliability and Safety (IJRS)*, 2(3), 209–240.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

Soria, J. L. A., Switzer, A. D., Pilarczyk, J. E., Tang, H., Weiss, R., Siringan, F., Manglicmot, M., Gallentes, A., Lau, A. A., Cheong, A. Y. L., et al. (2018). Surf beat-induced overwash during typhoon Haiyan deposited two distinct sediment assemblages on the carbonate coast of hernani, samar, central philippines. *Marine Geology*, 396, 215–230.

Soria, J. L. A., Switzer, A. D., Villanoy, C. L., Fritz, H. M., Bilgera, P. H. T., Cabrera, O. C., Siringan, F. P., Maria, Y. Y.-S., Ramos, R. D., & Fernandez, I. Q. (2016). Repeat storm surge disasters of typhoon Haiyan and its 1897 predecessor in the philippines. *Bulletin of the American Meteorological Society*, 97(1), 31–48.

Spiller, E. T., Bayarri, M., Berger, J. O., Calder, E. S., Patra, A. K., Pitman, E. B., & Wolpert, R. L. (2014). Automating emulator construction for geophysical hazard maps. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1), 126–152.

Sraj, I., Mandli, K. T., Knio, O. M., Dawson, C. N., & Hoteit, I. (2014). Uncertainty quantification and inference of Manning's friction coefficients using dart buoy data during the Tōhoku tsunami. *Ocean Modelling*, 83, 82–97.

Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. W. (2012). Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5), 3250–3265.

Srinivas, N., Krause, A., Seeger, M., & Kakade, S. M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 1015–1022).

Stefanakis, T. S., Contal, E., Vayatis, N., Dias, F., & Synolakis, C. (2014). Can small islands protect nearby coasts from tsunamis? An active experimental design approach. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 470(2172).

Steihaug, T. & Suleiman, S. (2013). Global convergence and the Powell singular function. *Journal of Global Optimization*, 56(3), 845–853.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media.

Takagi, H., Esteban, M., Shibayama, T., Mikami, T., Matsumaru, R., De Leon, M., Thao, N. D., Oyama, T., & Nakamura, R. (2017). Track analysis, simulation, and field survey of the 2013 typhoon Haiyan storm surge. *Journal of Flood Risk Management*, 10(1), 42–52.

Trehan, S., Carlberg, K., & Durlofsky, L. J. (2017). Error estimation for surrogate models of dynamical systems using machine learning. *arXiv preprint arXiv:1701.03240*.

Urban, N. M. & Fricker, T. E. (2010). A comparison of latin hypercube and grid ensemble designs for the multivariate emulation of an earth system model. *Computers & Geosciences*, 36(6), 746–755.

Vazquez, E. & Bect, J. (2007). Convergence properties of the expected improvement algorithm. *arXiv preprint arXiv:0712.3744*.

Vazquez, E. & Bect, J. (2010). Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and inference*, 140(11), 3088–3095.

Wang, C.-C., Lin, B.-X., Chen, C.-T., & Lo, S.-H. (2015). Quantifying the effects of long-term climate change on tropical cyclone rainfall using a cloud-resolving model: Examples of two landfall typhoons in taiwan. *Journal of Climate*, 28(1), 66–85.

Willem, L., Stijven, S., Vladislavleva, E., Broeckhove, J., Beutels, P., & Hens, N. (2014). Active learning to understand infectious disease models and improve policy making. *PLoS computational biology*, 10(4), e1003563.

Williams, C. A., Wedgwood, K. C., Mohammadi, H., Prouse, K., Tomlinson, O. W., & Tsaneva-Atanasova, K. (2019). Cardiopulmonary responses to maximal aerobic exercise in patients with cystic fibrosis. *PloS one*, 14(2), e0211219.

Williams, C. K. & Barber, D. (1998). Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12), 1342–1351.

Williams, C. K. I. & Rasmussen, C. E. (1996). Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8* (pp. 514–520). MIT Press.

Wilson, J., Hutter, F., & Deisenroth, M. (2018). Maximizing acquisition functions for bayesian optimization. In *Advances in Neural Information Processing Systems* (pp. 9884–9895).

Wu, J. & Frazier, P. (2016). The parallel knowledge gradient method for batch bayesian optimization. In *Advances in Neural Information Processing Systems* (pp. 3126–3134).

Yao, Y., He, F., Tang, Z., & Liu, Z. (2018). A study of tsunami-like solitary wave transformation and run-up over fringing reefs. *Ocean Engineering*, 149, 142–155.

Zhan, D., Qian, J., & Cheng, Y. (2017). Pseudo expected improvement criterion for parallel ego algorithm. *Journal of Global Optimization*, 68(3), 641–662.

Zhao, Z., Meza, J. C., & Hove, M. V. (2006). Using pattern search methods for surface structure determination of nanomaterials. *Journal of Physics: Condensed Matter*, 18(39), 8693.