Bayesian Numerical Analysis: Global Optimization and Other Applications

Jaroslav Fowkes
Exeter College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy in Numerical Analysis
Hilary Term, 2011

"In all our quest of greatness,
Like wanton boys, whose pastime is their care,
We follow after bubbles blown in th' air."

from The Duchess of Malfi
by John Webster

Abstract

We present a unifying framework for the global optimization of functions which are expensive to evaluate. The framework is based on a Bayesian interpretation of radial basis function interpolation which incorporates existing methods such as Kriging, Gaussian process regression and neural networks. This viewpoint enables the application of Bayesian decision theory to derive a sequential global optimization algorithm which can be extended to include existing algorithms of this type in the literature. By posing the optimization problem as a sequence of sampling decisions, we optimize a general cost function at each stage of the algorithm. An extension to multi-stage decision processes is also discussed.

The key idea of the framework is to replace the underlying expensive function by a cheap surrogate approximation. This enables the use of existing branch and bound techniques to globally optimize the cost function. We present a rigorous analysis of the canonical branch and bound algorithm in this setting as well as newly developed algorithms for other domains including convex sets. In particular, by making use of Lipschitz continuity of the surrogate approximation, we develop an entirely new algorithm based on overlapping balls.

An application of the framework to the integration of expensive functions over rectangular domains and spherical surfaces in low dimensions is also considered. To assess performance of the framework, we apply it to canonical examples from the literature as well as an industrial model problem from oil reservoir simulation.

Acknowledgements

First of all, I would like to thank my supervisors Nick Gould and Chris Farmer for introducing me to this fascinating field and for their continuing support and advice over the years. Researching this thesis has been an inspiring and thoroughly enjoyable experience which would not have been possible without their valuable input.

I am grateful to EPSRC and Schlumberger for providing the Industrial CASE studentship which supported the research in this thesis. Schlumberger have been particularly accommodating and provided access to their reservoir simulation software for the industrial application example.

Last but not least, I would like to thank everyone else who has helped to make this thesis a reality: you know who you are.

Table of Contents

List of Symbols

$f(x)$	objective function
$\mathcal{D}$	domain of the objective function
n	dimension of the domain $\mathcal{D}$
N	number of sample points of the objective function
$x_1, \ldots, x_N$	objective function sample points
$y = (y_1, \ldots, y_N)^T$	vector of objective function values at the sample points
y_{min}	minimum of $y_1, \ldots, y_N$
F	stochastic process associated with the objective function
$s(x)$	surrogate function (also the mean of F given y)
$e(x)$	error in the surrogate (also the standard deviation of F given y)
$\mathcal{P}(\cdot)$	probability measure for F
$\varphi(\cdot)$	radial basis function (also the correlation function of F)
R	matrix of correlations between $x_1, \ldots, x_N$
$r(x)$	vector of correlations between x and $x_1, \ldots, x_N$
λ	vector of coefficients of radial surrogate term
$\mathcal{N}_\varphi$	native space for $\varphi(\cdot)$
W	diagonal matrix of norm weights w_i
$\sigma^2\varphi(0)$	variance of F
Π_d^n	space of polynomials of degree less than d in $\mathbb{R}^n$
P	matrix containing the basis of Π_d^n evaluated at $x_1, \ldots, x_N$
$p(x)$	vector containing the basis of Π_d^n evaluated at x

μ	vector of coefficients of polynomial surrogate term
$l(x)$	cost function
$E[\cdot]$	expected value
$V[\cdot]$	variance
$\mathrm{Cov}[\cdot, \cdot]$	covariance
$\rho(\cdot)$	probability density function (pdf)
$\Phi(\cdot)$	standard normal cumulative distribution function
$\phi(\cdot)$	standard normal probability density function
I	identity matrix
g	generic gradient
H	generic Hessian

Chapter 1

Introduction

It is perhaps surprising just how many interesting and important scientific problems can be reformulated as the maximisation or minimisation of a real valued function. Examples include such diverse problems as predicting the structures of proteins, managing financial portfolios, modelling chemical processes and finding where to drill new oil wells, to name but a few. In many of these cases there is the additional difficulty that evaluating the underlying function expends a considerably amount of time or money. It is therefore meaningful to study the global optimization problem

$$\min_{x \in \mathcal{D}} f(x)$$

where $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ is an expensive objective function whose exact form may be unknown. By expensive we mean that evaluating the function at any point $x \in \mathcal{D}$ takes a significant amount of resources. For example, a single evaluation might require running a reservoir simulation model or performing a laboratory experiment. We usually assume that f is continuous and $\mathcal{D}$ is compact so that the global optimization problem is well posed and we can derive theoretical results such as convergence, although in practice this may not be the case. Since minimising f is equivalent to maximising $-f$, the convention will be to always formulate the optimization problem as a minimisation.

As the objective function f is essentially unknown, it is natural to view it as a random function, i.e. to model it as a stochastic process. This is particularly advantageous since the mean of the process then provides an inexpensive approximation $s(x)$, a surrogate as we shall refer to it, to the expensive objective function. Furthermore, the variance of the process $e^2(x)$ gives us a measure of error in the approximation which can be used to further improve the surrogate. Armed with a surrogate approximation of our objective function and associated stochastic interpretation, we then seek to use this to determine where to further sample the objective function so as to locate its global minimum. It is pertinent to

view this as a decision problem, which lends itself to an application of Bayesian decision theory and leads us to derive a global optimization framework which iteratively samples the objective function. We will show that our framework incorporates existing methods for global optimization using surrogates proposed in the literature as specific instances and can be extended to include non-decision theoretic approaches. This, however, is not the only application of the framework, it can also be used for integrating the objective function as we will demonstrate.

1.1 Global Optimization

Let us first briefly review existing approaches in the literature for the global optimization of general non-convex functions $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$. We say that a point $x^* \in \mathcal{D}$ is a *global minimiser of* f if $f(x^*) \leq f(x)$ for all $x \in \mathcal{D}$. Then $f(x^*)$ is the *global minimum of* f over $\mathcal{D}$, which may be attained at more than one point x^*. A *local minimiser of* f on the other hand is a point $x^* \in \mathcal{D}$ such that $f(x^*) \leq f(x)$ for all x in some neighbourhood of x^*, and $f(x^*)$ is then a *local minimum of* f. It is important to note that finding the global minimum of f is much more difficult than finding a local minimum, as we will see in Chapter 2. Most approaches in the literature to finding the global minimum fall into one of three categories: exact or complete methods (mostly based on branch and bound), heuristic or incomplete methods (mostly based on stochastic search) and surrogate based methods, each of which we will consider in turn. For comprehensive general references on global optimization we refer the interested reader to Horst and Pardalos (1995), Pardalos and Romeijn (2002), Pinter (1996) as well as the survey by Neumaier (2004) and extensive references therein.

Branch and Bound

The basic idea of branch and bound is to partition the domain $\mathcal{D}$ of the objective function f into subdomains $\mathcal{B}_1, ..., \mathcal{B}_K$ and determine a lower bound α_i and upper bound β_i on f over each subdomain $\mathcal{B}_i$. We then discard a subdomain $\mathcal{B}_i$ if $\alpha_i > \min\{\beta_1, ..., \beta_k\}$ and apply this procedure recursively to the remaining subdomains. That is, each remaining $\mathcal{B}_i$ is partitioned into subdomains, lower and upper bounds for each subdomain are computed and so on. The recursion is stopped when the smallest lower bound is sufficiently close (i.e. within some tolerance $\varepsilon > 0$) to the smallest upper bound. Once the algorithm terminates, one of the remaining subdomains will have an upper and lower bound within ε of the global minimum (see Section 12 of Neumaier, 2004, for more details). A notable problem with this approach is that one needs to be able to compute lower and upper bounds for the objective function on the subdomains. The upper bound β_i is often chosen to be the value of the objective function evaluated at some point in $\mathcal{B}_i$, but the lower bound is more difficult to obtain. Moreover, the performance of the method is dependent on the effectiveness of the branch and bound algorithm used. There is a danger that if subdomains are not discarded fast enough (i.e. the bounds are not sufficiently tight) the algorithm will be forced to search the entire domain.

If the objective function happens to be Lipschitz continuous (with Lipschitz constant L) we can construct lower bounding functions for f which can be minimised to obtain lower bounds. In particular, if the domain $\mathcal{D}$ is partitioned into n-dimensional rectangles $\mathcal{B}_1, ..., \mathcal{B}_K$, one can obtain a lower bound for $f(x)$ over $\mathcal{B}_i$ by minimising the lower bounding function $\ell_i(x) = f(c_i) - L\|x - c_i\|$ where c_i is some element of $\mathcal{B}_i$ (see Section 5.3 of Pardalos, Horst and Thoai, 1995). Of course, the difficulty with this approach is that one has to know the value of L beforehand. Other approaches to finding lower bounds include convex relaxation (see Androulakis, Maranas and Floudas, 1995) or interval analysis (see Section 11 of Neumaier, 2004) amongst others.

Reasonably good (partly heuristic) branch and bound methods which avoid having to specify Lipschitz constants exist in the literature. These include the DIRECT method of Jones, Perttunen and Stuckman (1993) which we use to find the optimal weights in Section 3.3 and MCS by Huyer and Neumaier (1999).

Stochastic Search and Multistart

Stochastic search methods are heuristic global optimization algorithms for the objective function, characterised by the following generic procedure:

0. Select initial data $\mathcal{I} = \{x_1, \ldots, x_K\} \subset \mathcal{D}$.

1. Until termination, repeat the following procedure:

   a) Select $\mathcal{S} \subset \mathcal{I}$ *at random, preferring elements with low objective function value.*

   b) Randomly generate new data $x_1^*, \ldots, x_R^*$ *from elements* x_k *in* $\mathcal{S}$*, i.e. apply some random procedure to generate new data based on elements in* $\mathcal{S}$.

   c) Add new data to $\mathcal{I}$*, remove unwanted elements.*

Important examples of stochastic search methods include sampling the domain at random, genetic algorithms and simulated annealing. We refer the interested reader to Spall (2003) for details. In general, stochastic search methods exhibit slow convergence as they fail to exploit any underlying structure in the objective function. For example, the Differential Evolution method of Storn and Price (1997) we compare against in Section 8.1 belongs to this class of methods.

Stochastic multistart methods on the other hand, attempt to find all local minima of the objective function by starting local optimizers from suitably chosen initial points. For reasons of efficiency, such methods strive to only start the local optimization once in the region of attraction of each minimum. This is done by means of clustering methods which attempt to identify clusters of points in the same region of attraction (see Chapter 5 of Törn and Žilinskas, 1989). Such methods however, have no convergence guarantees and so can miss the global optimum.

Surrogate based Global Optimization

Both branch and bound and especially stochastic search methods require a large number of evaluations of the objective function, making them unsuitable for functions which are expensive to evaluate. In order to globally optimize such expensive functions, most approaches in the literature focus on optimizing a surrogate approximation of the objective function or some other criterion based on the surrogate (see Jones, 2001, for a review). We will therefore focus predominantly on this approach in the thesis. These methods can be based on a Bayesian interpretation of the objective function (see Mockus, 1989 and Chapter 6 of Törn and Žilinskas, 1989) or a seemingly deterministic criterion (see Gutmann, 2001). We will show in Section 5.2 that the majority of these approaches have a Bayesian decision theoretic interpretation and therefore postpone a more thorough review until then.

1.2 Outline of the Thesis

The layout of the thesis is as follows. We begin in Chapter 2 by presenting existing theory on the convergence of global optimization algorithms which iteratively sample the objective function as our proposed global optimization framework falls into this class. Our contribution in this chapter is merely to simplify the metric space convergence proof in the stochastic setting to the case of a real valued objective function and thereby bring it more in line with the deterministic proof. In addition, we highlight established results which show that in general the global optimization problem cannot be efficiently solved. Moving on to Chapter 3, we present our framework for constructing the surrogate approximation $s(x)$ to the objective function, making use of radial basis functions as correlation functions for the stochastic process. This is a long established theory developed across many different fields. Our contribution is to draw together and unify for the first time (to our knowledge) the differing viewpoints from the Bayesian, geostatistical and numerical analysis literature and to generalise them to include radial basis functions with a weighted norm. In particular, this provides a stochastic interpretation of radial basis function interpolation which we use later on in our optimization framework.

In Chapter 4 we review approaches in the literature to choosing where to sample the objective function to begin with so that we can construct the initial surrogate approximation for our iterative global optimization framework. Our contribution here is to briefly compare the different techniques numerically and show the equivalence of maximal L_∞-norm designs to a decision theoretic approach. We then present our complete iterative global optimization framework in Chapter 5, cast in the language of decision theory as the problem of where to next sample the objective function. Our contribution is to reformulate some of the existing approaches to the problem in the language of decision theory, giving them a rigorous theoretical justification and further extending them to multiple stage decision processes. By considering non-decision theoretic cost functions we can also incorporate additional methods for surrogate global optimization from the literature within our framework. Chapter 6

contains our main contributions. Here we provide global optimization algorithms using the branch and bound technique for our iterative optimization framework on various domains of the objective function. Further, we extend the branch and bound approach to an entirely new algorithm which makes use of the local Lipschitz continuity of the surrogate approximation. In all cases, we provide rigorous convergence proofs of the algorithms under suitable assumptions.

As we have alluded to previously, we briefly look at applying our global optimization framework to the problem of integrating the objective function over various domains in Chapter 7, and compare it with the standard approach using Monte Carlo integration. Our contribution is to extend these ideas to the surface of a sphere. Chapter 8 contains numerical examples of our proposed global optimization methodology and algorithms as well as an example application in reservoir simulation, finding the optimal location of a new oil well. The curse of dimensionality naturally poses computational challenges for our methods and for this reason we restrict our numerical examples to less than ten spatial dimensions. Finally, we conclude and discuss possible future research directions in Chapter 9.

Throughout the thesis we follow the convention that any unreferenced theorems or derivations represent original contributions whereas referenced theorems and derivations are merely reformulations of established results.

Chapter 2

Convergence Theory

We begin in this chapter by presenting existing convergence theory for global optimization algorithms which sequentially sample the objective function. In particular, we give necessary and sufficient conditions for the convergence of such algorithms to the global minimum of any continuous objective function. The surrogate based global optimization algorithms we are interested in belong to this class of sequential sampling algorithms and therefore the convergence results in this chapter are directly applicable. We will also briefly outline that in general the global optimization problem cannot be solved efficiently.

2.1 Deterministic Algorithms

Let us first consider deterministic global optimization algorithms which sequentially sample the objective function $f\colon \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$. We first need some initial concepts and definitions, the following are based on Definitions 2.2 and 2.3 in Stephens and Baritompa (1998) and Section 1.2.1.2 in Törn and Žilinskas (1989).

Definition 2.1. We say that *local information* is a function L, defined on the cartesian product of the set of continuous functions on $\mathcal{D}$ and the set of all finite sequences in $\mathcal{D}$, such that for any two continuous functions f and g on $\mathcal{D}$, any finite sequence $\{x_k\}_{k=1}^N \subset \mathcal{D}$ and any closed set $\mathcal{C}$ in $\mathcal{D}$ containing $\{x_k\}_{k=1}^N$, if $f_{|\mathcal{C}} = g_{|\mathcal{C}}$ then $L(f, \{x_k\}_{k=1}^N) = L(g, \{x_k\}_{k=1}^N)$.

In particular, local information includes information depending on function values and limiting information at a finite number of sample points. For example, function or gradient samples are local information whereas a Lipschitz constant is not.

Definition 2.2. A *deterministic sequential sampling algorithm* $\mathcal{A}$ for a function $f\colon \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ is an optimization algorithm which generates a sequence of iterates $\{x_k\}_{k=1}^\infty \subset \mathcal{D}$

such that x_1 depends only on $\mathcal{D}$ and each x_k depends only on $L(f, \{x_i\}_{i=1}^{k-1})$.

Definition 2.3. We say that a sequential sampling algorithm for $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ with sequence of iterates $\{x_k\}_{k=1}^{\infty}$ *converges to the global minimum* of f (assuming it exists) if

$$\min_{1 \leq k \leq m} f(x_k) \to \min_{x \in \mathcal{D}} f(x) \text{ as } m \to \infty.$$

Note that the above definition of a sequential sampling algorithm also includes finite algorithms (which can be thought of as a sequence where the final iterate is repeated ad infinitum). Also, the case where several initial iterates $x_1, \ldots, x_N$ depend on $\mathcal{D}$ is included (since this can be thought of as x_1 depending on $\mathcal{D}$ and $x_2, \ldots, x_N$ depending on x_1). We are now in a position to state and prove the main theorem of this section (an extension of Theorem 1.3 from Törn and Žilinskas, 1989):

Theorem 2.1 (Törn and Žilinskas, 1989)**.** *Let $\mathcal{D} \subset \mathbb{R}^n$ be compact. Then a deterministic sequential sampling algorithm $\mathcal{A}$ converges to the global minimum of any continuous function on $\mathcal{D}$ iff its sequence of iterates (for any continuous function on $\mathcal{D}$) is everywhere dense in $\mathcal{D}$.*

Proof: Let $\overline{\mathcal{S}}$ denote the closure of a set $\mathcal{S}$. As $\mathcal{D}$ is compact we have that any continuous function on $\mathcal{D}$ attains its minimum on $\mathcal{D}$ and so the global minimum exists and is well defined. Suppose that $f : \mathcal{D} \to \mathbb{R}$ is continuous and that the sequence of iterates $\{x_k\}_{k=1}^{\infty}$ of $\mathcal{A}$ is everywhere dense in $\mathcal{D}$. Then $\overline{\{x_k\}_{k=1}^{\infty}} = \mathcal{D}$ and we have that $\min_{1 \leq k \leq m} f(x_k) \to \min_{x \in \overline{\{x_k\}}} f(x) = \min_{x \in \mathcal{D}} f(x)$ as $m \to \infty$.

Conversely, assume that $\mathcal{A}$ converges to the global minimum of any continuous function on $\mathcal{D}$. Suppose, for a contradiction, that its sequence of iterates $\{x_k\}_{k=1}^{\infty}$ for a continuous function f is not everywhere dense in $\mathcal{D}$. If no such f exist we are done, otherwise there exists $\varepsilon > 0$ and $x_0 \in \mathcal{D}$ such that $\|x_0 - x_k\| > \varepsilon$ for all k. We now construct a function g which agrees with our original function f outside of $\mathcal{B}_\varepsilon(x_0) := \{x : \|x_0 - x\| < \varepsilon\}$ and takes a value smaller than the global minimum of f at x_0. We begin by constructing the cone

$$c(x) = \min_{x \in \mathcal{D}} f(x) - \delta + \frac{\|x_0 - x\|}{\epsilon} \left(\max_{x \in \mathcal{D}} f(x) - \min_{x \in \mathcal{D}} f(x) + \delta \right),$$

where $\delta > 0$. Then the sequence of iterates of $\mathcal{A}$ for the functions $f(x)$ and $g(x) := \min\{f(x), c(x)\}$ are the same since they agree outside of $\mathcal{B}_\varepsilon(x_0)$ as according to our definition of $\mathcal{A}$, each x_k depends only on $L(f, \{x_i\}_{i=1}^{k-1}) = L(g, \{x_i\}_{i=1}^{k-1})$ outside of $\mathcal{B}_\varepsilon(x_0)$. However, $\min_{1 \leq k \leq m} g(x_k)$ does not converge to $g(x_0) = \min_{x \in \mathcal{D}} g(x)$ as $m \to \infty$ since $g(x_k) - g(x_0) \geq \delta$ for all k and this contradicts our initial assumption. □

Notably, any surrogate based global optimization algorithm which depends only on local information is a deterministic sequential sampling algorithm and therefore the above theorem is directly applicable.

2.2 Stochastic Algorithms

In practice, most surrogate based global optimization algorithms start by taking a set of initial samples of the domain (see Chapter 4). If we assume that these initial samples depend only on the domain in question (and thus are the same if we run the algorithm repeatedly on the same domain) then the algorithm is a sequential sampling algorithm and Theorem 2.1 applies. If, on the other hand, the initial samples are chosen at random each time we run the algorithm on the same domain, Theorem 2.1 does not apply and we need a result which takes this randomness into account. With this in mind, we introduce stochastic global optimization algorithms which sequentially sample the objective function. These are defined as follows (cf. Definition 2.4 in Stephens and Baritompa, 1998):

Definition 2.4. A *stochastic sequential sampling algorithm* $\mathcal{A}$ for a function $f\colon \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ is an optimization algorithm for which there is local information L and which generates a sequence of iterates $\{x_k\}_{k=1}^{\infty} \subset \mathcal{D}$ such that x_1 depends only on $\mathcal{D}$ and ω_1 and each x_k depends only on $L(f, \{x_i\}_{i=1}^{k-1})$ and ω_k, an instance of a random variable.

Note that $\{x_k\}_{k=1}^{\infty}$ is now a random variable. We give the stochastic version of Theorem 2.1 (a special case of Theorem 3.3 from Stephens and Baritompa, 1998):

Theorem 2.2 (Stephens and Baritompa, 1998)**.** *Let $\mathcal{D} \subset \mathbb{R}^n$ be compact. Then for any probability p and any stochastic sequential sampling algorithm $\mathcal{A}$,*

$$P(\mathcal{A} \text{ converges to the global minimum of } f) \geq p, \quad \forall f \in C(\mathcal{D})$$
$$\text{iff } P(x \in \overline{\{x_k\}}_f) \geq p, \qquad \forall x \in \mathcal{D}, \quad \forall f \in C(\mathcal{D}),$$

where $\{x_k\}_f$ denotes the sequence of iterates of $\mathcal{A}$ for a function f and $C(\mathcal{D})$ denotes the set of continuous functions on $\mathcal{D}$.

Thus a stochastic sequential sampling algorithm converges with probability one to the global minimum of any continuous function on $\mathcal{D}$ iff it samples a dense subset of $\mathcal{D}$ with probability one.

Proof: As $\mathcal{D}$ is compact we have that any continuous function on $\mathcal{D}$ attains its minimum on $\mathcal{D}$ and so the global minimum exists and is well defined. Suppose that $f\colon \mathcal{D} \to \mathbb{R}$ is continuous and that $P(x \in \overline{\{x_k\}}_f) \geq p$ for all $x \in \mathcal{D}$. Then it follows immediately that the probability that any global minimiser of f is an iterate or subsequential limit point of $\{x_k\}_f$ is greater than or equal to p, and hence the probability that $\mathcal{A}$ converges to the global minimum of f is greater than or equal to p.

Conversely, assume that $P(\mathcal{A}$ converges to the global minimum of any continuous function on $\mathcal{D}) \geq p$. Suppose, for a contradiction, that there exists $f \in C(\mathcal{D})$ and $x_0 \in \mathcal{D}$ such that $P(x_0 \in \overline{\{x_k\}}_f) < p$. Then, as the set of subsequential limit points of $\{x_k\}_f$ is a subset of $\overline{\{x_k\}}_f$, we have that $P(x_0$ is a subsequential limit point of $\{x_k\}_f) < p$.

Define the non-negative real-valued random variable $R := \inf\{\|x - x_0\| : x \in \{x_k\}_f, x \neq x_0\}$. Since $R = 0$ implies x_0 is a subsequential limit point of $\{x_k\}_f$ we have that $P(R = 0) <$

p. By right-hand continuity of the cumulative distribution function, there exists $\nu > 0$ such that $P(R \leq \nu) < p$. Thus, $P(\overline{\{x_k\}}_f \cap \mathcal{N} \neq \emptyset) < p$ where $\mathcal{N} := \{x : \|x - x_0\| < \nu,\, x \neq x_0\}$ is a punctured neighbourhood of x_0.[1] As $\mathcal{N} \neq \emptyset$ there exist $y_0 \in \mathcal{N}$ and $\varepsilon > 0$ such that $\overline{\mathcal{B}_\varepsilon(y_0)} \subset \mathcal{N}$ and so $P(\overline{\{x_k\}}_f \cap \overline{\mathcal{B}_\varepsilon(y_0)} \neq \emptyset) < p$. See Figure 2.1 below for an illustration of this part of the proof.

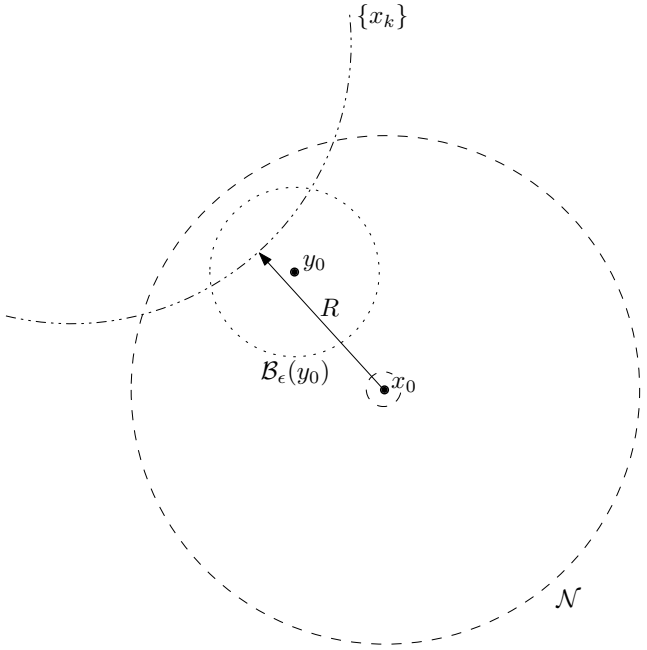

Figure 2.1: *An illustration of the proof of Theorem 2.2*

We now construct a function g which agrees with our original function f outside of $\mathcal{B}_\varepsilon(y_0)$ and takes a value smaller than the global minimum of f at y_0. We begin by constructing the cone

$$c(x) = \min_{x \in \mathcal{D}} f(x) - \delta + \frac{\|y_0 - x\|}{\varepsilon} \left(\max_{x \in \mathcal{D}} f(x) - \min_{x \in \mathcal{D}} f(x) + \delta \right),$$

where $\delta > 0$. Define $g(x) := \min\{f(x), c(x)\}$ and note that $\min_{x \in \mathcal{D}} g(x) = g(y_0) = \min_{x \in \mathcal{D}} f(x) - \delta$. Since according to our definition of $\mathcal{A}$, each x_k depends only on $L(f, \{x_i\}_{i=1}^{k-1})$ and ω_k it follows that for any realisation of $\{x_k\}_f$ where $\overline{\{x_k\}}_f \cap \overline{\mathcal{B}_\varepsilon(y_0)} = \emptyset$ we have that $\{x_k\}_f = \{x_k\}_g$. Thus $P(\overline{\{x_k\}}_g \cap \overline{\mathcal{B}_\varepsilon(y_0)} \neq \emptyset) < p$ and as the global minimiser of g is contained in $\mathcal{B}_\varepsilon(y_0)$ we have that the probability that it is an iterate or subsequential limit point of $\{x_k\}_g$ is less than p. We therefore conclude that $P(\mathcal{A}$ converges to the global minimum of $g) < p$, which contradicts our initial assumption. □

As in the deterministic case, any surrogate based global optimization algorithm which starts with a random initial sample (but otherwise depends only on local information) is a

[1] One may be tempted to replace the punctured neighbourhood $\mathcal{N}$ by a ball $\mathcal{B}_\nu(x_0)$ around x_0 and use it in place of the ball $\mathcal{B}_\varepsilon(y_0)$ in this stage of the proof, however this is not possible for the following reason: We would require $P(\overline{\{x_k\}}_f \cap \mathcal{B}_\nu(x_0) \neq \emptyset) < p$, but we have only proved $P(\overline{\{x_k\}}_f \cap \mathcal{B}_\nu(x_0) \setminus \{x_0\} \neq \emptyset) < p$. To see this observe that $R = \inf\{\|x - x_0\| : x \in \{x_k\}_f,\, x \neq x_0\} \leq \nu$ is only equivalent to $\overline{\{x_k\}}_f \cap \{x : \|x - x_0\| < \nu,\, x \neq x_0\} \neq \emptyset$. Indeed, $\overline{\{x_k\}}_f \cap \{x_0\} \neq \emptyset$ does to imply $R \leq \nu$ as we could have $x_k = x_0$ for some k.

stochastic sequential sampling algorithm and therefore the above theorem is directly applicable.

One can see from Theorems 2.1 and 2.2 that in order for our surrogate based global optimization algorithms to converge they need to sample a dense set (with probability one in the stochastic case). It should be noted however, that any sequential sampling algorithm can be modified to satisfy this condition by adding a random sample of the domain once every few thousand, say, iterations (this is similar to adding a gradient matching step in local optimization algorithms to ensure convergence to a critical point). Moreover, even random sampling of the domain (which is an extremely inefficient approach) converges with probability one. Thus convergence to the global minimum is not a suitable indicator of performance of surrogate optimization algorithms and one should think of it more as a necessary condition. One could think that perhaps proving termination of the algorithm in finite time is a better result, but termination in a million iterations is finite time yet highly unsatisfactory for expensive objective functions.

It should be noted that the sequence of iterates $\{x_k\}$ of the algorithms considered above does not necessarily converge to a global minimiser in the limit (in contrast to local optimization algorithms which, under sufficient assumptions, always converge to a critical point in the limit). One may well wonder whether it is at all possible to construct such an algorithm. The following theorems from Stephens and Baritompa (1998) provide an answer to this question:

Theorem 2.3. *For any deterministic sequential sampling algorithm, there exists a continuous function for which the sequence of iterates $\{x_k\}$ of the algorithm (or any subsequence thereof) does not converge to a global minimiser.*

Proof: See the proof of Theorem 3.2 in Stephens and Baritompa (1998). □

Theorem 2.4. *For any stochastic sequential sampling algorithm and $\varepsilon > 0$, there exists a continuous function for which the probability that the sequence of iterates $\{x_k\}$ of the algorithm (or any subsequence thereof) converges to a global minimiser is less than ε.*

Proof: See the proof of Theorem 3.4 in Stephens and Baritompa (1998). □

Thus, as surrogate optimisation algorithms generate a sequence of iterates which does not in general converge monotonically to the optimum, it is difficult to say what one means by a rate of convergence. Mockus (1994) suggests using the density ratio as a replacement for the rate of convergence. He defines the density ratio as the ratio of the density of samples in the vicinity of the optimum to the average density of samples. However, it is not clear how the density ratio is calculated in the general case and thus whether this is a suitable measure of convergence in general.

2.3 Complexity Theory

In this section we will briefly outline the computational complexity of the global optimization problem, that is to say how difficult it is to find the global minimum of a continuous objective function. Before we present the main result, we first need to introduce the class of NP-hard problems. The formal definition of the class of problems which are NP-hard is rather unwieldy so we will not state it here and instead refer the interested reader to Chapter 2 of Vavasis (1991). For our purposes it is sufficient to know that at present there does not exist an algorithm which can solve an NP-hard problem in polynomial time. By polynomial time we mean that the running time of the algorithm is bounded above by a polynomial expression in the size of the problem (here one can think of size as the dimension of the problem). Thus the key observation is that NP-hard problems cannot be solved efficiently.

With this in mind, let $\mathcal{F}$ be a set of functions $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ which is *closed*. By closed we mean that $\mathcal{F}$ contains linear functions and is closed under addition, multiplication by a positive constant and linear substitution, i.e. if $f(x) \in \mathcal{F}$ then $f(ax + b) \in \mathcal{F}$ for $a, b \in \mathbb{R}$. In particular, with this definition the set of linear functions and convex functions is closed. Further, let the term *box* denote an n-dimensional rectangle. We are now in a position to state the main result of this section (Theorem 2 from Kreinovich and Kearfott, 2005):

Theorem 2.5 (Kreinovich and Kearfott, 2005)**.** *Let $\varepsilon > 0$ and let $\mathcal{F}$ be a closed set of functions which contains at least one non-convex non-linear function. Then the problem of finding the global minimum of a function $f \in \mathcal{F}$ over a box $\mathcal{B} \subset \mathcal{D}$ to within an absolute accuracy ε is NP-hard.*

Proof: See the proof of Theorem 2 in Kreinovich and Kearfott (2005). □

In short, this theorem shows that in general the problem of finding the global minimum of f over a box is NP-hard and moreover that it can only be efficiently solved when f is convex. Thus in the worst case the time required to solve the global optimization problem over a box grows exponentially as function of the dimension n. But what about other regions? In particular, consider the n-dimensional ball. In this case if f is a quadratic function then the problem of finding the global minimum of f over a ball $\mathcal{B} \subset \mathcal{D}$ can be solved in polynomial time (see Section 4.3 of Vavasis, 1991). Thus, if instead we consider quadratic functions over a ball the global optimization problem can be solved efficiently. We will show in Section 6.5 how we can use this to our advantage when solving the global optimization problem over a box.

Chapter 3

Surrogate Framework

The key idea behind our approach to expensive function optimization is to approximate the underlying expensive objective function f by a function which is substantially cheaper to evaluate, called a surrogate (also referred to as an emulator or proxy in the reservoir simulation literature). Note that in the statistical analysis literature the term emulator refers to the entire distribution associated with f, not just the surrogate. We will use a radial basis function (RBF) interpolant as the surrogate since this provides us with a robust and rigorous interpolation framework. Over the years, RBF interpolation has become a well established and widely used approach to scattered data interpolation, particularly in higher dimensions (see Wendland, 2005). This approach is also known as intrinsic random function Kriging in the geostatistical literature (as proposed by Matheron, see Chilès and Delfiner, 1999) and Gaussian process regression in Bayesian statistics (an early reference is O'Hagan, 1978). Under these synonyms RBF interpolation has been applied to the design and analysis of computer experiments (Santner, Williams and Notz, 2003), machine learning (Rasmussen and Williams, 2006) and engineering design (Forrester, Sóbester and Keane, 2008) to name but a few. The RBF interpolant can also be viewed as a neural network (see Chapter 5 of Bishop, 1996). We refer the interested reader to the aforementioned books and references therein for extensive treatments of the underlying theory, albeit from very different perspectives.

Let us begin by defining the weighted ℓ_2-norm $\|\cdot\|_W := \|W\cdot\|_2$ with diagonal weight matrix W, and suppose we have N samples $y = (y_1, \ldots, y_N)^T$ of the objective function f at the corresponding sample points $x_1, \ldots, x_N \in \mathcal{D}$. The RBF surrogate is then constructed as a linear combination of basis functions $\varphi(\cdot)$ composed with a weighted ℓ_2-norm, together

with an additional polynomial term to guarantee uniqueness:

$$s(x) = \sum_{k=1}^{M} \mu_k \pi_k(x) + \sum_{j=1}^{N} \lambda_j \varphi(\|x - x_j\|_W) \tag{3.0.1}$$

where $\{\pi_k(x)\}_{k=1}^{M}$ is a basis for Π_d^n, the space of polynomials in $\mathbb{R}^n$ of degree less than d, with the notation $\Pi_0^n = \{0\}$. Typical choices of the basis function $\varphi(\cdot)$ are

$$\begin{aligned}
\text{the spline} \qquad & \varphi(r) = \begin{cases} r^p & \text{if } p \text{ is odd} \\ r^p \log r & \text{if } p \text{ is even} \end{cases}; \\
\text{the multiquadric} \qquad & \varphi(r) = (r^2 + \gamma^2)^{\beta} \qquad \beta > 0,\ \beta \notin \mathbb{N}; \\
\text{the inverse multiquadric} \qquad & \varphi(r) = (r^2 + \gamma^2)^{-\beta} \qquad \beta > 0; \text{ and} \\
\text{the Gaussian} \qquad & \varphi(r) = \exp(-\gamma^2 r^2),
\end{aligned} \tag{3.0.2}$$

where γ is a nonzero constant referred to as the shape parameter (see, for example, Chapters 6, 7, 8 of Wendland, 2005). As we use a weighted norm, we often let $\gamma = 1$ for the Gaussian basis function. The coefficients μ_k, λ_j are determined by solving the linear interpolation system

$$y_i = \sum_{k=1}^{M} \mu_k \pi_k(x_i) + \sum_{j=1}^{N} \lambda_j \varphi(\|x_i - x_j\|_W), \quad i = 1, \ldots, N$$

along with the additional conditions

$$\sum_{j=1}^{N} \lambda_j \pi_k(x_j) = 0, \quad k = 1, \ldots, M$$

which complete the system and ensure that polynomials of degree less than d are interpolated exactly. In matrix form this gives the non-singular (provided $\{x_i\}_{i=1}^{N}$ is a Π_d^n-unisolvent set, see Section 3.6) symmetric saddle-point system

$$\begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix} \tag{3.0.3}$$

where $P_{i,j} = \pi_j(x_i)$ is a polynomial basis matrix and R is the correlation matrix given by

$$R_{i,j} = \varphi(\|x_i - x_j\|_W).$$

We present two derivations of the RBF interpolation framework, a frequentist derivation often presented in the Universal Kriging literature (which in our case is equivalent to intrinsic random function Kriging) and an entirely Bayesian derivation from the statistical literature. First we need some initial concepts and assumptions which we draw predominantly from Chilès and Delfiner (1999). Recall that $\mathcal{D} \subset \mathbb{R}^n$ is the domain of our objective function and let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability space with sample space Ω, σ-algebra $\mathcal{F}$ and probability measure $\mathcal{P}$. A *stochastic process* (or *random function*) is then a function $F : \mathcal{D} \times \Omega \to \mathbb{R}$ such that for each $x \in \mathcal{D}$, $F(x, \cdot)$ is a random variable on $(\Omega, \mathcal{F}, \mathcal{P})$ (denoted by $F(x)$) and for

each $\omega \in \Omega$, $F(\cdot, \omega)$ is a function on $\mathcal{D}$ called a realisation of the stochastic process (denoted by $f(x)$). Furthermore, a stochastic process is called *Gaussian* if the distribution of any finite number of its random variables is multivariate normal. Now, suppose the objective function $f : \mathcal{D} \to \mathbb{R}$ is a realisation of a stochastic process F of the form

$$F(x) = \sum_{k=1}^{M} \mu_k \pi_k(x) + Z(x) \tag{3.0.4}$$

where the first term specifies its polynomial mean function and Z is another stochastic process which specifies the covariance structure. This will be done by means of a conditionally positive definite function of order d (see Definition 3.2), that is, a *generalised covariance function*. Such a function is only a covariance function for *generalised increments of order* d, which are linear combinations of the form $\sum_{i=1}^{L} a_i Z(x_i)$ for some x_i, a_i and L satisfying $\sum_{i=1}^{L} a_i \pi_k(x_i) = 0$ for all $k = 1, \ldots, M$. The process Z is assumed to have mean zero and covariance

$$\mathrm{Cov}[Z(x), Z(y)] = \sigma^2 \varphi(\|x - y\|_W) \tag{3.0.5}$$

between generalised increments $Z(x), Z(y)$ and therefore also specifies (3.0.5) as the generalised covariance function for F. We assume that F is Gaussian for generalised increments so that the mean and generalised covariance function are sufficient to completely specify the process. Then $\varphi(\|x - y\|_W)$ is the correlation function and $\sigma^2 \varphi(0)$ the variance of generalised increments of the process F. We choose the correlation function to be a radial basis function $\varphi(\cdot)$ composed with the weighted ℓ_2-norm

$$\|x\|_W^2 = \sum_{i=1}^{n} w_i^2 x_i^2$$

where w_i is the i-th diagonal entry of the diagonal matrix W. Since F has a polynomial mean function and its correlation function $\varphi(\|x - y\|_W)$ depends only on $x - y$, generalised d-th order increments of the process F have a covariance function which depends only on $x - y$ and have by definition zero mean. To see the latter note that

$$E\left[\sum_{i=1}^{L} a_i F(x_i)\right] = \sum_{i=1}^{L} a_i \sum_{k=1}^{M} \mu_k \pi_k(x_i) = \sum_{k=1}^{M} \mu_k \sum_{i=1}^{L} a_i \pi_k(x_i) = 0$$

by the definition of a generalised d-th order increment (here $E[\cdot]$ denotes statistical expectation in our probability space). A stochastic process S with constant mean function whose covariance function $\mathrm{Cov}[S(x), S(y)]$ depends only on $x - y$ (and not on x and y individually) is termed *second order stationary* and it therefore follows from the above that generalised d-th order increments of the process F are second order stationary. Furthermore, a process whose generalised d-th order increments are second order stationary is termed an *intrinsic random function of order d* and therefore F is an intrinsic random function of order d. The weighted norm means that in general the process F will be anisotropic. Given these preliminary assumptions we now proceed with the derivations where for the sake of brevity we will treat the generalised covariance function as an ordinary covariance function with the implicit assumption that we are always considering generalised increments.

3.1 Frequentist Derivation

Following Schonlau (1997) we derive the surrogate as the best linear unbiased predictor for the process F at untried $x \in \mathcal{D}$, as first proposed by Matheron (1962) for Kriging. For our scattered data samples $y = (y_1, \ldots, y_N)^T$ a linear predictor $s(x)$ for F is given by

$$s(x) = a^T(x)y$$

where $a(x)$ is an unknown vector of coefficients. Note that here we view y as a vector of random variables prior to sampling the objective function, i.e. $y_1 = F(x_1)$, etc. The best linear unbiased predictor is then obtained by minimising the mean squared error of prediction

$$E\Big[\big(a^T(x)y - F(x)\big)^2\Big] \tag{3.1.1}$$

with respect to $a(x)$ subject to the unbiasedness constraint

$$E\big[a^T(x)y - F(x)\big] = 0. \tag{3.1.2}$$

Since y is regarded as a realisation of F we have from (3.0.4) that

$$E\big[a^T(x)y - F(x)\big] = a^T(x)P\mu - p^T(x)\mu \tag{3.1.3}$$

where $P_{i,j} = \pi_j(x_i)$ is a polynomial basis matrix as before and $p(x) = (\pi_1(x), \ldots, \pi_M(x))^T$. Equating (3.1.2) and (3.1.3) for all μ we get that (3.1.2) can be rewritten as

$$P^T a(x) = p(x). \tag{3.1.4}$$

Thus the unbiasedness constraint (3.1.2) ensures that $a^T(x)y - F(x)$ is a generalised increment by enforcing (3.1.4), i.e. that $\sum_{i=1}^N a_i(x)\pi_k(x_i) - \pi_k(x) = 0$ for all $k = 1, \ldots, M$ (cf. the definition of a generalised increment on page 15). Now, for the mean squared error (3.1.1) we have

$$\begin{aligned}
E\Big[\big(a^T(x)y - F(x)\big)^2\Big] &= E\big[a^T(x)yy^Ta(x) + F^2(x) - 2a^T(x)yF(x)\big] \\
&= E\Big[a^T(x)(P\mu + z)(P\mu + z)^Ta(x) + \big(p^T(x)\mu + Z(x)\big)^2 \\
&\quad - 2a^T(x)(P\mu + z)\big(p^T(x)\mu + Z(x)\big)\Big] \\
&= E\Big[\big(a^T(x)P\mu\big)^2\Big] + E\Big[\big(a^T(x)z\big)^2\Big] + 2E\big[a^T(x)P\mu\, a^T(x)z\big] \\
&\quad + E\Big[\big(p^T(x)\mu\big)^2\Big] + E\Big[(Z(x))^2\Big] + 2E\big[p^T(x)\mu Z(x)\big] \\
&\quad - 2E\big[a^T(x)P\mu\, p^T(x)\mu\big] - 2E\big[a^T(x)z\, p^T(x)\mu\big] \\
&\quad - 2E\big[a^T(x)P\mu Z(x)\big] - 2E\big[a^T(x)zZ(x)\big] \\
&= \big(a^T(x)P\mu - p^T(x)\mu\big)^2 + a^T(x)\sigma^2Ra(x) + \sigma^2\varphi(0) - 2a^T(x)\sigma^2r(x) \\
&= \sigma^2\Big(a^T(x)Ra(x) - 2a^T(x)r(x) + \varphi(0)\Big)
\end{aligned} \tag{3.1.5}$$

where the second equality follows from (3.0.4), the penultimate from (3.0.5) and the last from (3.1.4). Here $z = (Z(x_1), \ldots, Z(x_N))^T$, $R_{i,j} = \varphi(\|x_i - x_j\|_W)$ is the correlation matrix

as before and $r(x) = (\varphi(\|x - x_1\|_W), \ldots, \varphi(\|x - x_N\|_W))^T$ is the cross-correlation vector. To minimise the mean squared error (3.1.5) with respect to (3.1.4) we introduce the vector of Lagrange multipliers $b(x)$ for the unbiasedness constraint (3.1.4) and the Lagrangian $\ell(a(x), b(x)) = \frac{1}{2}a^T(x)Ra(x) - a^T(x)r(x) + \frac{1}{2}\varphi(0) + b(x)\left(P^T a(x) - p(x)\right)$. Equating the derivative of the Lagrangian with respect to $a(x)$ to zero gives

$$Ra(x) - r(x) + Pb(x) = 0.$$

and similarly, equating the derivative of the Lagrangian with respect to $b(x)$ to zero gives

$$P^T a(x) - p(x) = 0.$$

Solving our minimisation problem thus leads to solving the linear system

$$Ra(x) + Pb(x) = r(x) \tag{3.1.6a}$$
$$P^T a(x) = p(x) \tag{3.1.6b}$$

or in matrix form

$$\begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} a(x) \\ b(x) \end{pmatrix} = \begin{pmatrix} r(x) \\ p(x) \end{pmatrix}. \tag{3.1.7}$$

The best linear unbiased predictor $s(x)$ is then given by

$$\begin{aligned} s(x) = a^T(x)y &= \begin{pmatrix} r(x) \\ p(x) \end{pmatrix}^T \begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} y \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} r(x) \\ p(x) \end{pmatrix}^T \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \quad \text{by (3.0.3)} \\ &= \sum_{k=1}^{M} \mu_k \pi_k(x) + \sum_{j=1}^{N} \lambda_j \varphi(\|x - x_j\|) \end{aligned}$$

which is the RBF interpolant with a weighted norm. As the predictor is unbiased its variance $e^2(x)$ is given by the mean squared error

$$\begin{aligned} e^2(x) &= E\Big[\big(a^T(x)y - F(x)\big)^2\Big] \\ &= \sigma^2\Big(a^T(x)Ra(x) - 2a^T(x)r(x) + \varphi(0)\Big) && \text{by (3.1.5)} \\ &= \sigma^2\Big(a^T(x)\left(r(x) - Pb(x)\right) - 2a^T(x)r(x) + \varphi(0)\Big) && \text{by (3.1.6a)} \\ &= \sigma^2\Big(-a^T(x)Pb(x) - a^T(x)r(x) + \varphi(0)\Big) \\ &= \sigma^2\Big(-p^T(x)b(x) - a^T(x)r(x) + \varphi(0)\Big) && \text{by (3.1.6b)} \\ &= \sigma^2\Big(\varphi(0) - \big(a^T(x)r(x) + b^T(x)p(x)\big)\Big) \\ &= \sigma^2\left[\varphi(0) - \begin{pmatrix} r(x) \\ p(x) \end{pmatrix}^T \begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} r(x) \\ p(x) \end{pmatrix}\right] && \text{by (3.1.7).} \end{aligned}$$

It now remains to determine the weights w_i^2 and the parameter σ^2. We will refer to these as covariance function parameters $\theta = (\sigma^2, W)$ and discuss how to determine them in Section 3.3. Finally, we note that RBF interpolation on a domain $\mathcal{D}$ with a weighted ℓ_2-norm

$$\|x\|_W^2 = \|Wx\|_2^2$$

is equivalent to standard RBF interpolation with the ℓ_2-norm on a scaled domain $\mathcal{S}$ where

$$\mathcal{S} := \{Wx \ : \ x \in \mathcal{D}\}$$

and thus all the usual RBF interpolation theory applies. In particular, the convergence and stability results from Section 3.6 hold in the case when a weighted norm is used, provided the weights are kept fixed.

3.2 Bayesian Derivation

We follow the derivation in Kitanidis (1986) due to O'Hagan (1978) for Gaussian process regression. Recall that there are N samples $y = (y_1, \ldots, y_N)^T$ of the objective function f at the corresponding sample points $x_1, \ldots, x_N \in \mathcal{D}$. Then from the form of the underlying Gaussian stochastic process F (3.0.4) we have

$$y = P\mu + z$$

where $P_{i,j} = \pi_j(x_i)$ as before and $z = (Z(x_1), \ldots, Z(x_n))^T$. We also have from (3.0.4) that

$$F(x) = p^T(x)\mu + Z(x)$$

where $p(x) = (\pi_1(x), \ldots, \pi_M(x))^T$ and $\mu = (\mu_1, \ldots, \mu_M)^T$. Thus $P\mu$ is the mean vector of y and $p^T(x)\mu$ the mean function of the process $F(x)$. Let $\tilde{R}$ denote the covariance matrix of y, $\tilde{r}(x)$ the cross-covariance vector between $F(x)$ and y and $\tilde{r}_0$ the variance of $F(x)$. Thus by our initial assumption (3.0.5) on the process covariance, $\tilde{R}_{i,j} = \sigma^2 R_{i,j}, \tilde{r}(x) = \sigma^2 r(x)$ and $\tilde{r}_0 = \sigma^2\varphi(0)$ where $R_{i,j} = \varphi(\|x_i - x_j\|_W)$ is the correlation matrix and $r(x) = (\varphi(\|x - x_1\|_W), \ldots, \varphi(\|x - x_N\|_W))^T$ the cross-correlation vector as in Section 3.1. Let $\theta = (\sigma^2, W)$ denote the covariance function parameters. For the purposes of this derivation θ is assumed known and therefore for notational convenience will be treated as background information and not explicitly denoted. The case where θ is unknown is treated in detail in Section 3.3. The probability distribution of $F(x)$ given μ and y (and θ), $\rho(F(x)|\mu, y)$ is then Gaussian with mean

$$E[F(x)|\mu, y] = p^T(x)\mu + \tilde{r}^T(x)\tilde{R}^{-1}(y - P\mu)$$

and variance

$$V[F(x)|\mu, y] = \tilde{r}_0 - \tilde{r}^T(x)\tilde{R}^{-1}\tilde{r}(x)$$

(see e.g. Rasmussen and Williams, 2006, equation (A.6), page 200). Let $\rho(\mu)$ denote the prior pdf of μ and $\rho(\mu|y)$ denote the posterior pdf of μ after observing the data y. Assume

that the prior distribution of μ is Gaussian with mean b and covariance matrix Σ. As y is Gaussian with mean $P\mu$ and covariance matrix $\tilde{R}$, the likelihood of μ (and θ) given the data, $\rho(y|\mu)$ is Gaussian with mean

$$E[y|\mu] = P\mu$$

and variance

$$V[y|\mu] = \tilde{R}.$$

We have from Bayes' theorem that the posterior pdf of μ after observing the data y is given by

$$\rho(\mu|y) \propto \rho(y|\mu)\rho(\mu). \tag{3.2.1}$$

Note that a Gaussian stochastic process has the defining property that the joint distribution of a finite number of random variables is multivariate normal. We therefore have that

$$\rho(y|\mu) = \frac{1}{(2\pi)^{N/2}\det(\tilde{R})^{1/2}} \exp\left(-\frac{1}{2}(y - P\mu)^T\tilde{R}^{-1}(y - P\mu)\right) \tag{3.2.2}$$

and

$$\rho(\mu) = \frac{1}{(2\pi)^{M/2}\det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(\mu - b)^T\Sigma^{-1}(\mu - b)\right). \tag{3.2.3}$$

Thus applying Bayes' theorem gives

$$\rho(\mu|y) \propto \rho(y|\mu)\rho(\mu) \propto \exp\left(-\frac{1}{2}(y - P\mu)^T\tilde{R}^{-1}(y - P\mu)\right) \exp\left(-\frac{1}{2}(\mu - b)^T\Sigma^{-1}(\mu - b)\right).$$

Letting $\Sigma^{-1} \to 0$ (i.e. letting $\rho(\mu)$ be a diffuse prior) and rearranging in terms of μ gives

$$\rho(\mu|y) \propto \exp\left(-\frac{1}{2}\left(\mu - \tilde{\Gamma}P^T\tilde{R}^{-1}y\right)^T P^T\tilde{R}^{-1}P\left(\mu - \tilde{\Gamma}P^T\tilde{R}^{-1}y\right)\right) \tag{3.2.4}$$

where $\tilde{\Gamma} = (P^T\tilde{R}^{-1}P)^{-1}$. Hence the posterior pdf of μ after observing the data y is Gaussian with mean

$$\gamma = \tilde{\Gamma}P^T\tilde{R}^{-1}y \tag{3.2.5}$$

and covariance matrix

$$\tilde{\Gamma} = (P^T\tilde{R}^{-1}P)^{-1}.$$

We can then calculate the compound distribution

$$\rho(F(x)|y) = \int_\mu \rho(F(x)|\mu, y)\rho(\mu|y)d\mu$$

which is Gaussian and has mean given by the law of total expectation (see Weiss, 2005, Proposition 10.6, page 599)

$$\begin{aligned} E[F(x)|y] &= E_\mu[E[F(x)|\mu, y]] \\ &= \int_\mu \left(p^T(x)\mu + \tilde{r}^T(x)\tilde{R}^{-1}(y - P\mu)\right)\rho(\mu|y)d\mu \\ &= \tilde{r}^T(x)\tilde{R}^{-1}y\int_\mu \rho(\mu|y)d\mu + \left(p^T(x) - \tilde{r}^T(x)\tilde{R}^{-1}P\right)\int_\mu \mu\rho(\mu|y)d\mu \end{aligned}$$

$$
\begin{aligned}
&= \tilde{r}^T(x)\tilde{R}^{-1}y + \left(p^T(x) - r^T(x)\tilde{R}^{-1}P\right)\gamma \\
&= p^T(x)\tilde{\Gamma}P^T\tilde{R}^{-1}y + \tilde{r}^T(x)\left(\tilde{R}^{-1} - \tilde{R}^{-1}P\tilde{\Gamma}P^T\tilde{R}^{-1}\right)y \\
&= \begin{pmatrix}\tilde{r}(x)\\ p(x)\end{pmatrix}^T \begin{pmatrix}\tilde{R}^{-1} - \tilde{R}^{-1}P\tilde{\Gamma}P^T\tilde{R}^{-1} & \left(\tilde{\Gamma}P^T\tilde{R}^{-1}\right)^T \\ \tilde{\Gamma}P^T\tilde{R}^{-1} & -\tilde{\Gamma}\end{pmatrix}\begin{pmatrix}y\\0\end{pmatrix} \\
&= \begin{pmatrix}\tilde{r}(x)\\ p(x)\end{pmatrix}^T \begin{pmatrix}\tilde{R} & P\\ P^T & 0\end{pmatrix}^{-1}\begin{pmatrix}y\\0\end{pmatrix} \\
&= \begin{pmatrix}\tilde{r}(x)\\ p(x)\end{pmatrix}^T \begin{pmatrix}\frac{\lambda}{\sigma^2}\\ \mu\end{pmatrix} \\
&= \sum_{k=1}^{M}\mu_k\pi_k(x) + \sum_{j=1}^{N}\lambda_j\varphi(\|x - x_j\|_W).
\end{aligned}
$$

where we have used the partitioned matrix inversion formula (see e.g. Rasmussen and Williams, 2006, equation (A.12), page 201). This is the RBF interpolant with a weighted norm and is the same as the best linear unbiased predictor $s(x)$ derived in Section 3.1. The compound distribution has variance given by the law of total variance (see Weiss, 2005, Proposition 10.7, page 600)

$$
\begin{aligned}
V[F(x)|y] &= E_\mu[V[F(x)|\mu, y]] + V_\mu[E[F(x)|\mu, y]] \\
&= \int_\mu V[F(x)|\mu, y]\rho(\mu|y)d\mu \\
&\quad + \int_\mu \left(E[F(x)|\mu, y] - E[F(x)|y]\right)\left(E[F(x)|\mu, y] - E[F(x)|y]\right)^T\rho(\mu|y)d\mu \\
&= \int_\mu \left(\tilde{r}_0 - \tilde{r}^T(x)\tilde{R}^{-1}\tilde{r}(x)\right)\rho(\mu|y)d\mu \\
&\quad + \int_\mu \left(p^T(x)\mu + \tilde{r}^T(x)\tilde{R}^{-1}(y - P\mu) - p^T(x)\gamma - \tilde{r}^T(x)\tilde{R}^{-1}(y - P\gamma)\right) \\
&\qquad \cdot \left(p^T(x)\mu + \tilde{r}^T(x)\tilde{R}^{-1}(y - P\mu) - p^T(x)\gamma - \tilde{r}^T(x)\tilde{R}^{-1}(y - P\gamma)\right)^T\rho(\mu|y)d\mu \\
&= \left(\tilde{r}_0 - \tilde{r}^T(x)\tilde{R}^{-1}\tilde{r}(x)\right)\int_\mu \rho(\mu|y)d\mu \\
&\quad + \left(p^T(x) - \tilde{r}^T(x)\tilde{R}^{-1}P\right)\int_\mu (\mu - \gamma)(\mu - \gamma)^T\rho(\mu|y)d\mu\left(p^T(x) - \tilde{r}^T(x)\tilde{R}^{-1}P\right)^T \\
&= \tilde{r}_0 - \tilde{r}^T(x)\tilde{R}^{-1}\tilde{r}(x) + \left(p^T(x) - \tilde{r}^T(x)\tilde{R}^{-1}P\right)\tilde{\Gamma}\left(p^T(x) - \tilde{r}^T(x)\tilde{R}^{-1}P\right)^T \\
&= \tilde{r}_0 + p^T(x)\tilde{\Gamma}p(x) - 2p^T(x)\tilde{\Gamma}P^T\tilde{R}^{-1}\tilde{r}(x) - \tilde{r}^T(x)\left(\tilde{R}^{-1} - \tilde{R}^{-1}P\tilde{\Gamma}P^T\tilde{R}^{-1}\right)\tilde{r}(x) \\
&= \sigma^2\Big[\varphi(0) + p^T(x)\Gamma p(x) - 2p^T(x)\Gamma P^TR^{-1}r(x) \\
&\qquad - r^T(x)\left(R^{-1} - R^{-1}P\Gamma P^TR^{-1}\right)r(x)\Big] \\
&= \sigma^2\left[\varphi(0) - \begin{pmatrix}r(x)\\ p(x)\end{pmatrix}^T\begin{pmatrix}R^{-1} - R^{-1}P\Gamma P^TR^{-1} & (\Gamma P^TR^{-1})^T\\ \Gamma P^TR^{-1} & -\Gamma\end{pmatrix}\begin{pmatrix}r(x)\\ p(x)\end{pmatrix}\right]
\end{aligned}
\tag{3.2.6}
$$

$$= \sigma^2 \left[\varphi(0) - \begin{pmatrix} r(x) \\ p(x) \end{pmatrix}^T \begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} r(x) \\ p(x) \end{pmatrix} \right]$$

where $\Gamma = (P^T R^{-1} P)^{-1}$. This variance is the same as the variance $e^2(x)$ of the best linear unbiased predictor derived in Section 3.1 and its square root (i.e. the standard deviation) can therefore be used as a measure of error in the surrogate fit. In fact, one can show for suitable objective functions f that the pointwise error $|f(x) - s(x)| \leq Ce(x)/\sigma$, where the positive constant C depends only on f (see Subsection 3.6.5).

Note that $e^2(x)$ is zero at any of the sample points x_i. This is indeed what one would expect since there is no uncertainty in the objective function at the points we have sampled. To see this, let $x = x_i$ and observe that $r(x_i)$ is then the i-th column of R. This means that

$$R^{-1} r(x_i) = u_i \quad \text{and} \quad r^T(x_i) R^{-1} = u_i^T \tag{3.2.7}$$

as R is symmetric, where u_i is the i-th unit vector. In particular, we get that

$$r^T(x_i) R^{-1} r(x_i) = r^T(x_i) u_i = \varphi(\|x_i - x_i\|_W) = \varphi(0). \tag{3.2.8}$$

Furthermore, $p^T(x_i)$ is the i-th row of P so that

$$P^T u_i = p(x_i) \quad \text{and} \quad u_i^T P = p^T(x_i). \tag{3.2.9}$$

We then have from (3.2.6) that

$$\begin{aligned} e^2(x_i) = \sigma^2 \Big[& \varphi(0) + p^T(x_i)\Gamma p(x_i) - 2p^T(x_i)\Gamma P^T R^{-1} r(x_i) \\ & - r^T(x_i) \left(R^{-1} - R^{-1} P \Gamma P^T R^{-1} \right) r(x_i) \Big] \\ = \sigma^2 & \left[\varphi(0) + p^T(x_i)\Gamma p(x_i) - 2p^T(x_i)\Gamma P^T u_i - \varphi(0) + u_i^T P \Gamma P^T u_i \right] \quad \text{by (3.2.7), (3.2.8)} \\ = \sigma^2 & \left[p^T(x_i)\Gamma p(x_i) - 2p^T(x_i)\Gamma p(x_i) + p^T(x_i)\Gamma p(x_i) \right] \quad \text{by (3.2.9)} \\ = 0. & \end{aligned}$$

It remains to determine the covariance function parameters $\theta = (\sigma^2, W)$ which is discussed in the next section.

3.3 Parameter Estimation

There are various approaches one can use to find the covariance function parameters $\theta = (\sigma^2, W)$ and we will focus on three in particular. These are all plug-in estimation methods which ignore uncertainty in the resulting estimates. It is possible to use fully Bayesian methods which integrate out the covariance function parameters but these are computationally more expensive (we refer the interested reader to Kitanidis, 1986, for more details). The first approach we consider is maximum likelihood estimation (MLE) and consists of choosing the parameters to maximise the likelihood of θ given the observed data y (see Section

3.3.2 of Santner et al., 2003). We have according to equation (3.2.2), that the likelihood of $\theta = (\sigma^2, W)$ and μ given the data y is

$$\rho(y|\mu, \sigma^2, W) = \frac{1}{(2\pi)^{N/2}\sigma^N \det(R)^{1/2}} \exp\left(-\frac{1}{2\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\right).$$

Hence, the log-likelihood is

$$l(\mu, \sigma^2, W|y) = -\frac{1}{2}\left(N\log(2\pi) + N\log\sigma^2 + \log\det(R) + \frac{1}{\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\right)$$

which can be expanded as

$$l(\mu, \sigma^2, W|y) = -\frac{1}{2}\Big(N\log(2\pi) + N\log\sigma^2 + \log\det(R) + \frac{1}{\sigma^2}\big(y^T R^{-1} y - y^T R^{-1} P\mu - \mu^T P^T R^{-1} y + \mu^T P^T R^{-1} P\mu\big)\Big).$$

Assuming the weights w_i^2 are fixed, differentiating with respect to μ and equating to zero we get that

$$\begin{aligned} 0 = \frac{\partial l(\mu, \sigma^2)}{\partial \mu} &= -\frac{1}{2\sigma^2}\left(-(R^{-1}P)^T y - P^T R^{-1} y + 2P^T R^{-1} P\mu\right) \\ &= -\frac{1}{\sigma^2}\left(-P^T R^{-1} y + P^T R^{-1} P\mu\right) \end{aligned}$$

as R is symmetric, which implies that

$$P^T R^{-1} P\mu = P^T R^{-1} y.$$

Hence, the maximum likelihood estimate of μ is given by

$$\mu = (P^T R^{-1} P)^{-1} P^T R^{-1} y \tag{3.3.1}$$

which is equivalent to the μ obtained by solving the linear interpolation system (3.0.3). Alternatively, this can be obtained directly as the posterior mean of μ (see equation (3.2.5)). Either way, substituting the estimate for μ into the log-likelihood, we get that the log-likelihood of $\theta = (\sigma^2, W)$ given the data y is

$$l(\sigma^2, W|y) = -\frac{1}{2}\left(N\log(2\pi) + N\log\sigma^2 + \log\det(R) + \frac{1}{\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\right)$$

where μ is now given by (3.3.1). Similarly, assuming the weights w_i^2 are fixed, differentiating the log-likelihood with respect to σ^2 and equating to zero we get that

$$0 = \frac{\partial l(\sigma^2)}{\partial \sigma^2} = -\frac{1}{2}\left(\frac{N}{\sigma^2} - \frac{1}{\sigma^4}\Big((y - P\mu)^T R^{-1}(y - P\mu)\Big)\right)$$

which implies that

$$N = \frac{1}{\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu).$$

Hence, the maximum likelihood estimate of σ^2 is given by

$$\sigma^2 = \frac{1}{N}(y - P\mu)^T R^{-1}(y - P\mu). \tag{3.3.2}$$

Substituting the maximum likelihood estimate for σ^2 back into the log-likelihood and ignoring constant terms gives

$$l(W) = -\frac{1}{2}\left(N \log \sigma^2 + \log \det(R)\right)$$

where $R_{i,j} = \varphi(\|x_i - x_j\|_W)$ is the correlation matrix. We maximise $l(W)$ using a global optimization algorithm (DIRECT, Jones et al., 1993) to obtain the weights w_i^2. For our C++ code, we use the NLopt implementation of the DIRECT algorithm. In practice it is helpful to place an upper bound on the value of the weights w_i^2 so as to ensure smoothness of the resulting surrogate. For this reason, and because we are predominantly interested in relative scaling in each coordinate direction, we restrict w_i^2 to the unit interval once the range of sample points x_i has been suitably scaled (see Subsection 3.6.3).

The second approach to choosing the covariance function parameters θ we consider is restricted maximum likelihood estimation (REML) which is considered preferable in the case where we have a mean term in the surrogate (see Santner et al., 2003). Following Kitanidis (1986), we will present a Bayesian derivation of the restricted maximum likelihood approach. In this approach the covariance function parameters are chosen to maximise the *marginal* likelihood of θ given the observed data y. Here we mean marginal in the sense of marginalising out μ. We assumed in Section 3.2 that the covariance function parameters θ were known in advance, but this is of course not the case. To remedy this, let $\rho(\theta)$ be a prior on θ whose form we will not, for the moment, specify. As before, let the probability distribution of μ given θ, $\rho(\mu|\theta)$ be Gaussian with mean $b(\theta)$ and covariance matrix $\Sigma(\theta)$, see (3.2.3). Then we have that the joint probability distribution of μ and θ is given by

$$\rho(\mu, \theta) = \rho(\mu|\theta)\rho(\theta) = \frac{1}{(2\pi)^{M/2}\det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(\mu - b(\theta))^T \Sigma(\theta)^{-1}(\mu - b(\theta))\right) \rho(\theta)$$

simply by the definition of conditional probability. We also have from earlier (see (3.2.2)) that the likelihood of θ and μ given the data y is

$$\rho(y|\mu, \theta) = \frac{1}{(2\pi)^{N/2}\sigma^N \det(R)^{1/2}} \exp\left(-\frac{1}{2\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\right).$$

Bayes' theorem then tells us that the joint probability distribution of μ and θ given the data y is

$$\begin{aligned}
\rho(\mu, \theta|y) &\propto \rho(y|\mu, \theta)\rho(\mu, \theta) \\
&\propto \frac{1}{(2\pi)^{N/2}\sigma^N \det(R)^{1/2}} \exp\left(-\frac{1}{2\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\right) \\
&\quad \cdot \frac{1}{(2\pi)^{M/2}\det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(\mu - b(\theta))^T \Sigma(\theta)^{-1}(\mu - b(\theta))\right) \rho(\theta).
\end{aligned}$$

As before, letting $\Sigma^{-1} \to 0$ so that we have a diffuse prior on μ, we get

$$\rho(\mu, \theta|y) \propto \frac{1}{(2\pi)^{N/2}\sigma^N \det(R)^{1/2}} \exp\left(-\frac{1}{2\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\right) \rho(\theta).$$

We also have from (3.2.4) that the probability distribution of μ given θ and y is

$$\rho(\mu|\theta, y) \propto \frac{1}{(2\pi)^{M/2}\sigma^M \det(\Gamma)^{1/2}} \exp\left(-\frac{1}{2\sigma^2}\left(\mu - \Gamma P^T R^{-1} y\right)^T \Gamma^{-1} \left(\mu - \Gamma P^T R^{-1} y\right)\right)$$

where $\Gamma = (P^T R^{-1} P)^{-1}$. The posterior marginal probability distribution of θ is then

$$\begin{aligned}\rho(\theta|y) = \frac{\rho(\mu, \theta|y)}{\rho(\mu|\theta, y)} \propto & \frac{1}{(2\pi)^{(N-M)/2}\sigma^{N-M}\det(R)^{1/2}\det(\Gamma)^{-1/2}} \\ & \cdot \exp\left(-\frac{1}{2\sigma^2} y^T \left(R^{-1} - R^{-1} P \Gamma P^T R^{-1}\right) y\right) \rho(\theta)\end{aligned}$$

which after some elementary algebra can be rewritten in the more familiar form

$$\rho(\theta|y) \propto \frac{1}{(2\pi)^{(N-M)/2}\sigma^{N-M}\det(R)^{1/2}\det(\Gamma)^{-1/2}} \exp\left(-\frac{1}{2\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\right) \rho(\theta)$$

where μ is now given by $\mu = \Gamma P^T R^{-1} y$ as before. The marginal likelihood $\rho(y|\theta)$ is therefore

$$\begin{aligned}\rho(y|\theta) \propto & \frac{1}{(2\pi)^{(N-M)/2}\sigma^{N-M}\det(R)^{1/2}\det(\Gamma)^{-1/2}} \\ & \cdot \exp\left(-\frac{1}{2\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\right)\end{aligned} \tag{3.3.3}$$

and the marginal log-likelihood is then, up to a constant, given by

$$\begin{aligned}l(\sigma^2, W|y) = -\frac{1}{2}\Bigg(& (N - M)\log(2\pi) + (N - M)\log\sigma^2 + \log\det(R) \\ & + \log\det(P^T R^{-1} P) + \frac{1}{\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu)\Bigg).\end{aligned}$$

since $\theta = (\sigma^2, W)$. As before, assuming the weights w_i^2 are fixed, differentiating the marginal log-likelihood with respect to σ^2 and equating to zero we get that

$$0 = \frac{\partial l(\sigma^2)}{\partial \sigma^2} = -\frac{1}{2}\left(\frac{N - M}{\sigma^2} - \frac{1}{\sigma^4}\Big((y - P\mu)^T R^{-1}(y - P\mu)\Big)\right)$$

which implies that

$$N - M = \frac{1}{\sigma^2}(y - P\mu)^T R^{-1}(y - P\mu).$$

Hence, the restricted maximum likelihood estimate of σ^2 is given by

$$\sigma^2 = \frac{1}{N - M}(y - P\mu)^T R^{-1}(y - P\mu). \tag{3.3.4}$$

Substituting the restricted maximum likelihood estimate for σ^2 back into the marginal log-likelihood and ignoring constant terms gives

$$l(W) = -\frac{1}{2}\left((N - M)\log\sigma^2 + \log\det(R) + \log\det(P^T R^{-1} P)\right)$$

where $R_{i,j} = \varphi(\|x_i - x_j\|_W)$, which we again maximise using a global optimization algorithm to obtain the weights w_i^2 restricted to the unit interval. If we now specify a diffuse prior on θ (i.e. let $\rho(\theta) \propto 1$), the likelihood is equivalent to the posterior and thus the restricted maximum likelihood estimate for θ derived above is the same as the fully Bayesian maximum a posteriori estimate. Maximum a posteriori estimation (MAP) is yet another parameter estimation technique at our disposal but we do not consider it in great detail here as in our case it is equivalent to restricted maximum likelihood estimation. This would of course not be the case if we were to specify informative priors on θ and μ. In such a setting one could also marginalise the distribution of $F(x)|\theta, \mu, y$ over θ and μ using Monte Carlo integration as pursued by Osborne, Garnett and Roberts (2009).

The final approach we consider is leave-one-out cross-validation (LOOCV) which leads one to choose the parameters θ to minimise the ℓ_2-norm of the cross-validation error (see Rippa, 1999). As we already have an analytical maximum likelihood estimate (3.3.2) or (3.3.4) for σ^2, we will only use leave-one-out cross-validation to find the weights w_i^2 for reasons of computational efficiency. However, like maximum likelihood estimation the technique can be applied to find any number of parameters. We will extended the derivation found in Rippa (1999) to the case of surrogates with polynomial terms. First, we define some notation. Let the superscript $v^{[t]}$ denote the vector v with the t-th component removed and $M^{[t]}$ the matrix M with the t-th row and column removed. Recall that we have samples $y_1, \ldots, y_N$ of the objective function $f(x)$ at sample points $x_1, \ldots, x_N \in \mathcal{D}$. Let

$$s^t(x) = \sum_{k=1}^{M} \mu_k \pi_k(x) + \sum_{\substack{j=1 \\ j \neq t}}^{N} \lambda_j^{[t]} \varphi(\|x - x_j\|_W)$$

be the RBF interpolant to $y^{[t]} = (y_1, \ldots, y_{t-1}, y_{t+1}, \ldots, y_N)^T$ at $x^{[t]} = (x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_N)^T$ and let

$$\epsilon_t(W) = y_t - s^t(x_t) \quad \text{for } t = 1, \ldots, N$$

be the error at the left out validation point x_t. Also recall that

$$A = \begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix} \tag{3.3.5}$$

is the usual RBF interpolation matrix from our interpolation system (3.0.3) given by

$$A \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix} \tag{3.3.6}$$

which is uniquely solvable (see Theorem 3.3). Consider the following system

$$Aa = e_t \tag{3.3.7}$$

where e_t is the t-th unit vector and note that $a_t \neq 0$. (For if $a_t = 0$ then as $A^{[t]}a^{[t]} = 0$ this would imply that $a = 0$, which would contradict the fact that a solves the above non-singular

system.) Define

$$b = \begin{pmatrix} \lambda \\ \mu \end{pmatrix} - \frac{\lambda_t}{a_t} a. \tag{3.3.8}$$

Then premultiplying (3.3.8) by A and using (3.3.6), (3.3.7) gives

$$\begin{aligned} Ab &= A \begin{pmatrix} \lambda \\ \mu \end{pmatrix} - \frac{\lambda_t}{a_t} Aa \\ &= \begin{pmatrix} y \\ 0 \end{pmatrix} - \frac{\lambda_t}{a_t} e_t \\ &= \left(y_1, \ldots, y_{t-1}, y_t - \frac{\lambda_t}{a_t}, y_{t+1}, \ldots, y_N, 0, \ldots, 0\right)^T \end{aligned} \tag{3.3.9}$$

and as $b_t = 0$ by construction (3.3.8) we get from (3.3.9) that

$$A^{[t]} b^{[t]} = \begin{pmatrix} y^{[t]} \\ 0 \end{pmatrix}. \tag{3.3.10}$$

Hence by uniqueness of the solution to the linear systems (3.3.6) and (3.3.10) we have that

$$\begin{aligned} \lambda^{[t]} &= \left(b_1, \ldots, b_{t-1}, b_{t+1}, \ldots, b_N\right)^T, \\ \mu &= \left(b_{N+1}, \ldots, b_{N+M}\right)^T. \end{aligned}$$

This implies that

$$\begin{aligned} s^t(x_t) &= \sum_{k=1}^{M} \mu_k \pi_k(x_t) + \sum_{\substack{j=1 \\ j \neq t}}^{N} \lambda_j^{[t]} \varphi(\|x_t - x_j\|_W) \\ &= \sum_{k=1}^{M} b_{k+N} \pi_k(x_t) + \sum_{\substack{j=1 \\ j \neq t}}^{N} b_j \varphi(\|x_t - x_j\|_W) \\ &= \sum_{k=1}^{M} b_{k+N} \pi_k(x_t) + \sum_{j=1}^{N} b_j \varphi(\|x_t - x_j\|_W) \\ &= (Ab)_t \\ &= y_t - \frac{\lambda_t}{a_t} \end{aligned}$$

where the third line follows from the fact that $b_t = 0$ and the last line from (3.3.9). Thus the t-th element of the cross-validation error $\epsilon(W)$ is given by

$$\epsilon_t(W) = y_t - s^t(x_t) = \frac{\lambda_t}{a_t}.$$

Finally, observe that a_t is the t-th diagonal element of A^{-1} since a is the t-th column of A^{-1} by construction (3.3.7). Summarising, we find that W is chosen to minimise the ℓ_2-norm of the cross-validation error $\epsilon(W)$ defined componentwise as

$$\epsilon_t(W) = \frac{\lambda_t}{A^{-1}_{t,t}} \text{ for } t = 1, \ldots, N.$$

and depends on W through the correlation matrix $R_{i,j} = \varphi(\|x_i - x_j\|_W)$ in the matrix A, see (3.3.5). As before, the minimisation is done using a global optimization algorithm with the weights w_i^2 restricted to the unit interval.

3.4 Incorporating Derivatives

If derivatives of the objective function $f(x)$ are available and not prohibitively expensive to obtain (e.g. from an adjoint code) they can be used to improve the accuracy of the surrogate approximation $s(x)$, provided the objective function is not highly oscillatory. Incorporating derivatives for oscillatory objective functions typically worsens the accuracy of the surrogate approximation since this emphasises local oscillations as opposed to the more global trends we are interested in (see example below).

Suppose the objective function $f(x)$ is sufficiently smooth so that first order derivatives are available (it is possible to incorporate higher order derivatives but the method is analogous so we will not discuss it further). Recall that we view the objective function $f(x)$ as a realisation of a stochastic process F with mean function given by $\mu^T p(x)$ and covariance function given by (3.0.5). The stochastic process specifies derivative processes (see Papoulis, 1991, Appendix 10A, for details on existence)

$$\frac{\partial F(x)}{\partial x^a}, \quad a = 1, \ldots, n$$

with mean functions given by

$$E\left[\frac{\partial F(x)}{\partial x^a}\right] = \mu^T \frac{\partial}{\partial x^a} p(x), \quad a = 1, \ldots, n \tag{3.4.1}$$

and covariance functions by

$$\mathrm{Cov}\left[\frac{\partial F(x)}{\partial x^a}, \frac{\partial F(y)}{\partial y^b}\right] = \sigma^2 \frac{\partial^2 \varphi(\|x - y\|_W)}{\partial x^a \partial y^b}, \quad a, b = 1, \ldots, n \tag{3.4.2}$$

where x^a denotes the a-th component of x in $\mathbb{R}^n$. Now, suppose we have samples of $f(x)$ and its derivatives at the points $x_1, \ldots, x_N \in \mathcal{D}$ in the vector $\bar{y} = (\hat{y}(x_1), \ldots, \hat{y}(x_N))^T$ where $\hat{y}(x) = \left(f(x), \frac{\partial f(x)}{\partial x^1}, \ldots, \frac{\partial f(x)}{\partial x^n}\right)$. Following through the derivation in Section 3.2 with the samples $\bar{y}$ and the relevant mean and covariance matrices defined by (3.4.1),(3.4.2) one can show that the posterior mean of the process F (i.e. the surrogate) is given by

$$E[F(x)|\bar{y}] = \sum_{k=1}^{M} \mu_k \pi_k(x) + \sum_{j=1}^{N} \lambda_j^0 \varphi(\|x - x_j\|_W) + \sum_{j=1}^{N} \sum_{a=1}^{n} \lambda_j^a \frac{\partial \varphi(\|x - x_j\|_W)}{\partial x^a} \tag{3.4.3}$$

where the coefficients $\bar{\lambda} = (\lambda_1^0, \ldots, \lambda_1^n, \ldots, \lambda_N^0, \ldots, \lambda_N^n)^T$ and μ are obtained by solving the symmetric saddle-point system

$$\begin{pmatrix} \bar{R} & \bar{P} \\ \bar{P}^T & 0 \end{pmatrix} \begin{pmatrix} \bar{\lambda} \\ \mu \end{pmatrix} = \begin{pmatrix} \bar{y} \\ 0 \end{pmatrix}. \tag{3.4.4}$$

Here $\bar{R}$ is the derivative correlation matrix given by

$$\bar{R} = \begin{pmatrix} \hat{R}(x_1, x_1) & \dots & \hat{R}(x_1, x_N) \\ \vdots & & \vdots \\ \hat{R}(x_N, x_1) & \dots & \hat{R}(x_N, x_N) \end{pmatrix},$$

where

$$\hat{R}(x,y) = \begin{pmatrix} \varphi(\|x-y\|_W) & \frac{\partial\varphi(\|x-y\|_W)}{\partial x^1} & \dots & \frac{\partial\varphi(\|x-y\|_W)}{\partial x^n} \\ \frac{\partial\varphi(\|x-y\|_W)}{\partial y^1} & \frac{\partial^2\varphi(\|x-y\|_W)}{\partial x^1\partial y^1} & \dots & \frac{\partial^2\varphi(\|x-y\|_W)}{\partial x^n\partial y^1} \\ \vdots & \vdots & & \vdots \\ \frac{\partial\varphi(\|x-y\|_W)}{\partial y^n} & \frac{\partial^2\varphi(\|x-y\|_W)}{\partial x^1\partial y^n} & \dots & \frac{\partial^2\varphi(\|x-y\|_W)}{\partial x^n\partial y^n} \end{pmatrix}$$

is the submatrix of derivative correlations between x and y. Similarly, $\bar{P}$ is the polynomial basis derivative matrix given by

$$\bar{P} = \left(\hat{P}(x_1), \dots, \hat{P}(x_N)\right)^T, \quad \text{where } \hat{P}(x) = \left(p(x), \frac{\partial p(x)}{\partial x^1}, \dots, \frac{\partial p(x)}{\partial x^n}\right)$$

and $p(x) = (\pi_1(x), \dots, \pi_M(x))^T$ as before. One can similarly derive an analogous expression for the posterior variance of the process F

$$V[F(x)|\bar{y}] = \sigma^2 \left[\varphi(0) - \begin{pmatrix} \bar{r}(x) \\ p(x) \end{pmatrix}^T \begin{pmatrix} \bar{R} & \bar{P} \\ \bar{P}^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \bar{r}(x) \\ p(x) \end{pmatrix}\right]$$

where $\bar{r}(x) = \left(\hat{r}_1(x), \dots, \hat{r}_N(x)\right)^T$ is the derivative cross-correlation vector with

$$\hat{r}_j(x) = \left(\varphi(\|x-x_j\|_W), \frac{\partial\varphi(\|x-x_j\|_W)}{\partial x^1}, \dots, \frac{\partial\varphi(\|x-x_j\|_W)}{\partial x^n}\right).$$

Alternatively, the above can be obtained using the frequentist derivation from Section 3.1 (see Morris, Mitchell and Ylvisaker, 1993 and Forrester, Sóbester and Keane, 2007 for details).

If the objective function is highly oscillatory the above approach tends to yield inaccurate surrogates as the oscillations are over-emphasised. In an attempt to overcome this, one might be tempted to replace the derivatives in the surrogate approximation (3.4.3) and linear system (3.4.4) by finite differences with a judiciously chosen step length h. For example, one could use forward finite differences

$$\frac{\partial\varphi(\|x-x_j\|_W)}{\partial x^a} \approx \frac{\varphi(\|x+he_a-x_j\|_W) - \varphi(\|x-x_j\|_W)}{h}$$

where e_a is the a-th unit vector. However, this is equivalent to observing the objective function at extra points as one can readily see by substituting the above approximation into (3.4.3). As an example of a highly oscillatory function, consider fitting a surrogate with derivatives to the Ackley function (see Ackley, 1987). Figure 3.1 compares the surrogate fitted using first derivatives and function samples only to the original function. It is evident that in this case using function samples without derivatives gives a much better surrogate.

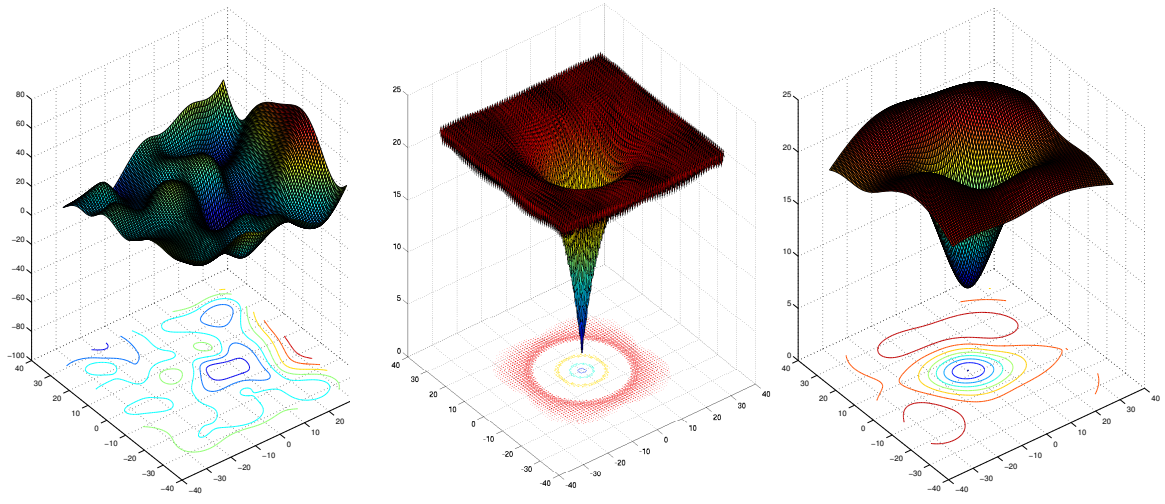

Figure 3.1: *Ackley function (centre) and its surrogate approximation using derivatives (left) and function samples only (right).*

3.5 Other Extensions

So far we have tacitly assumed that we can evaluate the objective function $f(x)$ exactly, that is to say, there is no measurement error when sampling the function. There are however, many situations where this is not the case. For example, if evaluating our objective function corresponds to running a simulation model of an oil reservoir, the value we obtain will often be uncertain due to truncation error in the simulator. In such cases the measurement error ϵ is usually assumed to be additive and iid normally distributed with mean zero and unknown variance σ_ϵ^2. The form of the underlying Gaussian stochastic process F (3.0.4) can then be generalised to include the measurement error

$$F(x) = \sum_{k=1}^{M} \mu_k \pi_k(x) + Z(x) + \epsilon.$$

The covariance (3.0.5) then becomes

$$\mathrm{Cov}[F(x), F(y)] = \sigma^2 \varphi(\|x - y\|_W) + \sigma_\epsilon^2 \delta_{xy}$$

where δ_{xy} is a Kronecker delta (which is zero unless $x = y$, in which case it is one). The unknown error variance σ_ϵ^2 is usually determined along with the other covariance function parameters θ (see Section 3.3). We refer the interested reader to Section 2.2 in Rasmussen and Williams (2006) for more details on incorporating measurement error.

Another possible extension is to handle data from multiple code levels as described in Kennedy and O'Hagan (2000) and Forrester et al. (2007), an approach known as co-kriging in the geostatistical literature (see e.g. Ver Hoef and Cressie, 1993). For example, running an oil reservoir simulator with different sized simulation grids will typically produce different

outputs. Suppose we have s such outputs arranged in order of increasing accuracy from the functions $f_1(x), \ldots, f_s(x)$. The form of the underlying stochastic process F (3.0.4) can then be generalised via the process Z to the different code levels

$$Z(x) = \begin{cases} Z_1(x) & \text{for the least accurate level 1} \\ Z_k(x) = \tau_{k-1} Z_{k-1}(x) + Z_k^d(x) & \text{for levels } k = 2, \ldots, s \end{cases}$$

where τ_{k-1} is a scaling parameter and Z_k^d is independent of $Z_{k-1}, \ldots, Z_1$. Each of the processes Z_k^d has mean zero and the usual covariance function

$$\mathrm{Cov}[Z_k^d(x), Z_k^d(y)] = \sigma_k^{d2} \varphi(\|x - y\|_W) \quad k = 1, \ldots, s$$

as does the least accurate code level Z_1

$$\mathrm{Cov}[Z_1(x), Z_1(y)] = \sigma_1^2 \varphi(\|x - y\|_W).$$

Note that we can determine the covariance function parameters σ_k^d, σ_1 and the scaling parameters τ_{k-1} independently for each level as in Section 3.3 due to the independence condition on Z_k^d.

3.6 Interpolation Theory

In this section we will look at the solvability and stability of the RBF interpolation system (3.0.3) as well as estimates of the rate of convergence of the surrogate $s(x)$ given by (3.0.1) to the objective function, drawing predominantly on established results from Wendland (2005). We will also briefly mention update strategies for the interpolation matrix $\left(\begin{smallmatrix} R & P \\ P^T & 0 \end{smallmatrix}\right)$.

3.6.1 Solvability of the Interpolation System

First of all we will show that the RBF interpolation system (3.0.3) is solvable for our choices of radial basis function (3.0.2). We begin with the following definitions (cf. Definitions 2.6 and 8.1 from Wendland, 2005):

Definition 3.1. The set of points $\{x_1, \ldots, x_N\} \subset \mathbb{R}^n$ is called a *Π_d^n-unisolvent set* if the zero polynomial is the only polynomial from Π_d^n that vanishes on all the points $x_1, \ldots, x_N$.

Note that this is equivalent to saying that $\mathrm{rank}(P) = M$ and therefore a Π_d^n-unisolvent set has at least M elements (see Remark 3.5 in Gutmann, 2001).

Definition 3.2. A continuous function $\Phi : \mathbb{R}^n \to \mathbb{R}$ is said to be *conditionally positive definite of order d* if for any $N \in \mathbb{N}$, for all pairwise distinct points $x_1, \ldots, x_N \in \mathbb{R}^n$ and for all $\lambda \in \mathbb{R}^N \setminus \{0\}$ satisfying

$$P^T \lambda = 0$$

where $P_{i,j} = \pi_j(x_i)$, $\{\pi_1(x), \ldots, \pi_M(x)\}$ is a basis of Π_d^n, we have that

$$\lambda^T R \lambda > 0,$$

where $R_{i,j} = \Phi(x_i - x_j)$.

Note that in general R will not be positive definite, however at least $N - M$ of its eigenvalues will be positive. There are several different approaches to characterising conditional positive definiteness of radial basis functions, but by far the most powerful is due to the following theorem (Theorem 2.1 in Micchelli, 1986). We first give a definition and then state the theorem.

Definition 3.3. A function $\psi \in C[0,\infty) \cap C^\infty(0,\infty)$ is said to be *strictly completely monotone on* $(0,\infty)$ if

$$(-1)^l \psi^{(l)}(r) > 0$$

for all $r > 0$ and $l \in \mathbb{N}_0$.

Theorem 3.1 (Micchelli, 1986)**.** *Let $\varphi \in C[0,\infty) \cap C^\infty(0,\infty)$. Then the function $\Phi := \varphi(\|\cdot\|_W)$ is conditionally positive definite of order $d \in \mathbb{N}_0$ on $\mathbb{R}^n$ for any $n \in \mathbb{N}$ if and only if $(-1)^d \varphi^{(d)}$ is strictly completely monotone on $(0,\infty)$.*

Proof: See the proof of Theorem 2.1 in Micchelli (1986). □

We then have the following corollary on the conditional positive definiteness of the standard radial basis functions (3.0.2) (cf. Corollary 8.20 in Wendland, 2005):

Corollary 3.2. *The following functions $\Phi : \mathbb{R}^n \to \mathbb{R}$ are conditionally positive definite of order d on $\mathbb{R}^n$ for any $n \in \mathbb{N}$:*

- *Odd Splines* $\Phi(x) = (-1)^{(p+1)/2} \|x\|_W^p$, *$p$ odd with* $d = (p+1)/2$
- *Even Splines* $\Phi(x) = (-1)^{p/2+1} \|x\|_W^p \log \|x\|_W$, *$p$ even with* $d = p/2 + 1$
- *Multiquadric* $\Phi(x) = (-1)^{\lceil \beta \rceil} (\|x\|_W^2 + \gamma^2)^\beta$, $\beta > 0$, $\beta \notin \mathbb{N}$ *with* $d = \lceil \beta \rceil$
- *Inverse Multiquadric* $\Phi(x) = (\|x\|_W^2 + \gamma^2)^{-\beta}$, $\beta > 0$ *with* $d = 0$
- *Gaussian* $\Phi(x) = \exp(-\gamma^2 \|x\|_W^2)$ *with* $d = 0$

where γ is a nonzero constant.

Proof: A simple application of Theorem 3.1, see proof of Corollary 8.20 in Wendland (2005). □

Note that the powers of -1 are absorbed into the coefficients for simplicity in our formulation (3.0.2). We are now in a position to state and prove the main theorem of this section (cf. Theorem 8.21 in Wendland, 2005):

Theorem 3.3. *Let $\Phi = \varphi(\|\cdot\|_W)$ be conditionally positive definite of order d and $\{x_1, \ldots, x_N\} \subset \mathbb{R}^n$ be a Π_d^n-unisolvent set. Then the RBF interpolation system* (3.0.3)

$$\begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}$$

is uniquely solvable.

Proof: Let $(\lambda, \mu)^T$ be in the null space of the RBF interpolation matrix $\left(\begin{smallmatrix} R & P \\ P^T & 0 \end{smallmatrix}\right)$. Then

$$R\lambda + P\mu = 0 \tag{3.6.1a}$$

$$P^T\lambda = 0 \tag{3.6.1b}$$

Note that (3.6.1b) is equivalent to the assumption in Definition 3.2. Multiplying (3.6.1a) by λ^T on the left gives $0 = \lambda^T R\lambda + \lambda^T P\mu = \lambda^T R\lambda$ by (3.6.1b). Therefore, as Φ is conditionally positive definite of order d we must have from Definition 3.2 that $\lambda = 0$. Hence, (3.6.1a) reduces to $P\mu = 0$ and since $\{x_1, \ldots, x_N\}$ is Π_d^n-unisolvent we have that $\mu = 0$. Thus the null space of the RBF interpolation matrix is trivial and so the matrix is nonsingular, which means that the RBF interpolation system is uniquely solvable. □

Note that the unisolvency condition is only required for uniqueness and not for solvability. Also, if we augment the system with low-dimensional polynomials, unisolvency is a relatively mild condition on the data.

3.6.2 Asymptotic Convergence Estimates

We will now give some results on the rate with which our surrogate $s(x)$ given by (3.0.1) converges to certain types of objective function f as the number of sample points tends to infinity, i.e. the asymptotic convergence rate. Throughout this section we assume that the weight matrix W is fixed. Let $\varphi(\cdot)$ be a conditionally positive definite radial basis function of order d. Following Gutmann (2001), we define the linear space

$$F_\varphi(\mathcal{D}) := \left\{ \sum_{i=1}^N \lambda_i \varphi(\|\cdot - x_i\|_W) : N \in \mathbb{N}, \lambda \in \mathbb{R}^N, x_1, \ldots, x_N \in \mathcal{D}, \sum_{i=1}^N \lambda_i p(x_i) = 0, p \in \Pi_d^n \right\}$$

which with the inner product

$$\left\langle \sum_{i=1}^N \lambda_i \varphi(\|\cdot - x_i\|_W), \sum_{j=1}^L \mu_j \varphi(\|\cdot - y_j\|_W) \right\rangle_\varphi := \sum_{i=1}^N \sum_{j=1}^L \lambda_i \mu_j \varphi(\|x_i - y_j\|_W)$$

becomes an inner product space. We can extend $F_\varphi(\mathcal{D})$ to the linear space

$$\mathcal{A}_\varphi(\mathcal{D}) = F_\varphi(\mathcal{D}) + \Pi_d^n$$

which when equipped with $< \cdot, \cdot >_\varphi$ becomes a semi-inner product space. Moreover, the semi-inner product induces the semi-norm $\|\cdot\|_\varphi := < \cdot, \cdot >_\varphi^{1/2}$. We then have the following characterisation of the surrogate s (cf. Theorem 4 in Schaback, 1993):

Theorem 3.4 (Schaback, 1993)**.** *Let φ be a conditionally positive definite radial basis function of order d. Suppose further that $\mathcal{D} \subset \mathbb{R}^n$ and that $\{x_1, \ldots, x_N\} \subset \mathcal{D}$ is a given Π_d^n-unisolvent set along with objective function values $f(x_1), \ldots, f(x_N) \in \mathbb{R}$. Then the surrogate s given by (3.0.1) is the unique element of $\mathcal{A}_\varphi(\mathcal{D})$ that minimises the semi-norm $\|g\|_\varphi$ over all functions $g \in \mathcal{A}_\varphi(\mathcal{D})$, subject to the interpolation conditions $g(x_i) = f(x_i)$ for all $i = 1, \ldots, N$.*

Proof: See the proof of Theorem 3.7 in Gutmann (2001) or Theorem 4 in Schaback (1993). □

One can further extend the linear space $\mathcal{A}_\varphi(\mathcal{D})$ to a larger native space $\mathcal{N}_\varphi(\mathcal{D})$ such that the surrogate $s \in \mathcal{A}_\varphi(\mathcal{D})$ to given data minimises the semi-norm $\|g\|_\varphi$ over all functions $g \in \mathcal{N}_\varphi(\mathcal{D})$ subject to the same interpolation conditions. Then for any function $f \in \mathcal{N}_\varphi(\mathcal{D})$, the surrogate $s \in \mathcal{A}_\varphi(\mathcal{D})$ to f at any Π_d^n-unisolvent set of points $\{x_1, \ldots, x_N\}$ has a semi-norm bounded above by $\|f\|_\varphi$. Hence a function is in the native space iff all of its surrogates have a uniformly bounded semi-norm. We can therefore define the native space $\mathcal{N}_\varphi(\mathcal{D})$ as follows (cf. Definition 3.10 in Gutmann, 2001):

Definition 3.4. The *native space* $\mathcal{N}_\varphi(\mathcal{D})$ of a conditionally positive definite radial basis function φ of order d is the space of functions $f\colon \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ such that for any Π_d^n-unisolvent set $\{x_1, \ldots, x_N\} \subset \mathcal{D}$ the surrogate $s \in \mathcal{A}_\varphi(\mathcal{D})$ to f at these points (given by (3.0.1)) satisfies

$$\|s\|_\varphi \leq C,$$

where the constant C depends only on f. The native space is a complete semi-inner product space with semi-inner product $< \cdot, \cdot >_{\mathcal{N}_\varphi(\mathcal{D})}$ and induced semi-norm $\|\cdot\|_{\mathcal{N}_\varphi(\mathcal{D})} := < \cdot, \cdot >^{1/2}_{\mathcal{N}_\varphi(\mathcal{D})}$. When $d = 0$ the native space becomes a Hilbert space.

The native spaces of the spline radial basis functions are Beppo Levi spaces (also called homogeneous Sobolev spaces) whereas the native spaces of the multiquadric, inverse multiquadric and Gaussian radial basis functions are rather small (see Chapter 10 in Wendland, 2005). Essentially these are spaces of very smooth functions. Before moving on to the asymptotic convergence estimates, we need some preliminary concepts and definitions. For a domain $\mathcal{D} \subset \mathbb{R}^n$ and sample points $x_1, \ldots, x_N \in \mathcal{D}$ we define the *fill distance* h as

$$h = \max_{x \in \mathcal{D}} \min_{1 \leq i \leq N} \|x - x_i\|_W.$$

We also define for a multi-index $\alpha = (\alpha_1, \ldots, \alpha_n)^T \in \mathbb{N}_0^n$ with length $|\alpha| = \alpha_1 + \ldots + \alpha_n$ and a sufficiently smooth function $f : \mathbb{R}^n \to \mathbb{R}$ the derivative

$$D^\alpha f(x) := \frac{\partial^{|\alpha|} f(x)}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}}$$

where $x_1, \ldots, x_n$ denote the components of x. Finally, we say that $\mathcal{D} \subset \mathbb{R}^n$ *satisfies an interior cone condition* if there exists $\theta \in (0, \pi/2), r > 0$ such that for every $x \in \mathcal{D}$ there exists a unit vector $\nu(x)$ such that the cone

$$C = \{x + ty : y \in \mathbb{R}^n, \|y\|_W = 1, y^T \nu(x) \geq \cos\theta, t \in [0, r]\}$$

is contained in $\mathcal{D}$. We are now in a position to state the main theorems of this section on the convergence rates of different radial basis functions (cf. Chapter 11 of Wendland, 2005):

Theorem 3.5. *Let φ be a multiquadric, inverse multiquadric or Gaussian radial basis function and suppose that it is conditionally positive definite of order d. Let $\mathcal{D} \subset \mathbb{R}^n$ be bounded and satisfy an interior cone condition and let $s(x)$ be one of the aforementioned radial basis function surrogates to a function $f \in \mathcal{N}_\varphi(\mathcal{D})$. Then for $\alpha \in \mathbb{N}_0^n$ and for any $l \in \mathbb{N}$ with $l \geq \max\{|\alpha|, d-1\}$ there exist $h_0(l), C_l > 0, C_l$ independent of h, such that*

$$|D^\alpha f(x) - D^\alpha s(x)| \leq C_l h^{l-|\alpha|} \|f\|_{\mathcal{N}_\varphi(\mathcal{D})}$$

for all $x \in \mathcal{D}$ provided $h \leq h_0(l)$.

Proof: See the proof of Theorem 11.14 in Wendland (2005). □

We have from Theorem 3.5 that for a multiquadric, inverse multiquadric or Gaussian basis function the RBF surrogate and all its derivatives (with multi-index α) converge with order $O(h^{l-|\alpha|})$ for any $l \geq \max\{|\alpha|, d-1\}$ to any function (or any of its derivatives) in the native space. However, if one considers only convergence of the function (and not the derivatives) in the L_∞-norm it is possible to prove a stronger result:

Theorem 3.6. *Let $\mathcal{D}$ be a hypercube in $\mathbb{R}^n$ and let $s(x)$ be a multiquadric, inverse multiquadric or Gaussian radial basis function surrogate of a function $f \in \mathcal{N}_\varphi(\mathcal{D})$. Then there exist $c > 0, c$ independent of h, such that*

$$\|f - s\|_{L_\infty(\mathcal{D})} \leq e^{-c/h} \|f\|_{\mathcal{N}_\varphi(\mathcal{D})}$$

for the multiquadric and inverse multiquadric and

$$\|f - s\|_{L_\infty(\mathcal{D})} \leq e^{c \log h/h} \|f\|_{\mathcal{N}_\varphi(\mathcal{D})}$$

for the Gaussian, whenever h is sufficiently small.

Proof: Follows directly from Theorem 11.22 in Wendland (2005). □

Thus for the multiquadric, inverse multiquadric or Gaussian basis functions we have spectral convergence of the interpolant to any function in the native space. A similar, albeit weaker result to Theorem 3.5 is true for the spline basis functions:

Theorem 3.7. *Let $\mathcal{D} \subset \mathbb{R}^n$ be bounded and satisfy an interior cone condition and let $s(x)$ be a spline ($\varphi(r) = r^p$ or $r^p \log r$) radial basis function surrogate of a function $f \in \mathcal{N}_\varphi(\mathcal{D})$. Then there exist $h_0, C > 0, C$ independent of h, such that*

$$|D^\alpha f(x) - D^\alpha s(x)| \leq C h^{p/2-|\alpha|} \|f\|_{\mathcal{N}_\varphi(\mathcal{D})}$$

for all $x \in \mathcal{D}$ and for any α with $|\alpha| \leq p/2 - 1$ provided $h \leq h_0$.

Proof: See the proof of Theorems 11.16 and 11.19 in Wendland (2005). □

We therefore have that for a spline basis function the RBF surrogate and all its derivatives (with multi-index α) converge with order $O(h^{p/2-|\alpha|})$ to any function (or any of its derivatives) in the native space (here p is the power of the radial term in the basis function). While this is a weaker convergence result than for the previous basis functions, we will see in the next section that spline basis functions lead to more stable interpolation matrices. In the case of an expensive function the asymptotic convergence rates are never realised in practice since one cannot afford the large number of function samples this would require. It therefore makes sense to focus on the stability of the interpolation process.

3.6.3 Stability of the Interpolation System

The stability of the radial basis function interpolation process depends primarily on the stability of the underlying interpolation system (3.0.3) which we have to solve to construct the surrogate. To ensure the interpolation matrix $\left(\begin{smallmatrix} R & P \\ P^T & 0 \end{smallmatrix}\right)$ is not ill-conditioned as a result of containing very large and very small entries, we scale the range of the data points x_i for each dimension. This also enables us to place an upper bound on the weights w_i^2 (see Section 3.3). In order to examine the stability of the interpolation system further we define

$$\lambda_{\min}(R) = \inf_{\lambda \in \mathbb{R}^n \setminus \{0\}, P^T\lambda = 0} \frac{\lambda^T R \lambda}{\lambda^T \lambda} = \inf_{v \in \mathbb{R}^{N-M}} \frac{v^T V^T R V v}{v^T v}$$

and similarly for $\lambda_{\max}(R)$ where $R_{i,j} = \varphi(\|x_i - x_j\|_W)$ is the RBF correlation matrix, $P_{i,j} = \pi_j(x_i)$ is the polynomial basis matrix and V is such that its columns span the null space of P^T, see (3.6.2). Note that by the conditional definiteness property (Definition 3.2) we have that $\lambda_{\min}(R) > 0$. We also define the *separation distance* q as

$$q = \frac{1}{2} \min_{i \neq j} \|x_i - x_j\|_W.$$

We can interpret the ratio $\kappa(R) := \lambda_{\max}(R)/\lambda_{\min}(R)$ as the condition number of the interpolation system (if the radial basis function is conditionally positive definite of order 0, this is precisely the condition number of the interpolation matrix in (3.0.3) as in this case $P = 0$). It can be shown (see Chapter 12 of Wendland, 2005) that for the general case $\lambda_{\max}(R)$ grows at most like $1/q^n$ if the data is quasi-uniformly distributed and so we are predominantly interested in the behaviour of $\lambda_{\min}(R)$. In particular, one can derive lower bounds for $\lambda_{\min}(R)$ for our standard radial basis functions (3.0.2) of the form

$$\lambda_{\min}(R) \geq C_\varphi(q)$$

where $C_\varphi(q)$ for the different radial basis functions is given (up to a constant) in Table 3.1 overleaf.

We refer the interested reader to Chapter 12 of Wendland (2005) for derivations of the above bounds. In particular, notice that for Gaussians and (inverse) multiquadrics the bound on $\lambda_{\min}(R)$ grows exponentially as a function of the separation distance q and so we might expect that interpolating with these radial basis functions would lead to ill-conditioned

Radial Basis Function φ	$C_\varphi(q)$
Splines $\varphi(r) = r^p$ or $r^p \log r$	q^p
(Inverse) Multiquadric $\varphi(r) = (r^2 + \gamma^2)^\beta$	$q^{\beta-(n-1)/2} e^{-12.76n\gamma/q}$
Gaussian $\varphi(r) = e^{-\gamma^2 r^2}$	$q^{-n} e^{-40.71n^2/(\gamma q)^2}$

Table 3.1: *Lower bounds for* $\lambda_{\min}(R)$

interpolation matrices. This is exactly what is observed in practice and we illustrate this behaviour by calculating the condition number $\kappa(R)$ of the interpolation system for the Gaussian $\varphi(r) = \exp(-r^2)$, inverse multiquadric $\varphi(r) = (r^2+1)^{-1/2}$ and cubic spline $\varphi(r) = r^3$ RBFs in three dimensions as the number of sample points increases (see Figure 3.2). The

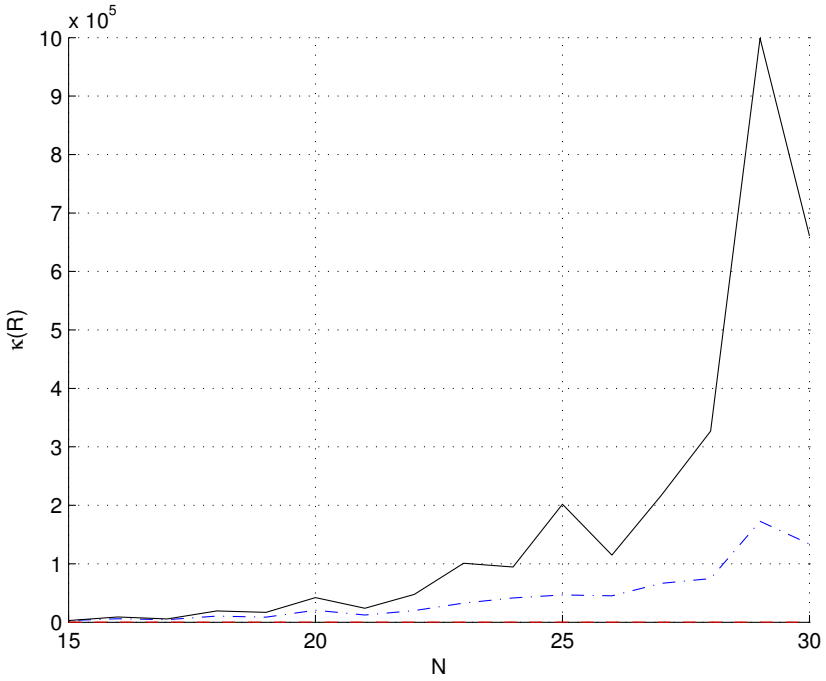

Figure 3.2: *A plot of the 'condition number'* $\kappa(R)$ *of the interpolation system for a Gaussian (solid black), inverse multiquadric (dash-dotted blue) and cubic spline (dashed red) RBF in three dimensions against the number* N *of maximin Latin hypercube sample points* $x_1, \ldots, x_N$ *(see Section 4.2).*

most common way of dealing with the ill-conditioning of the interpolation system is to use Tikhonov regularisation: instead of solving the linear interpolation system (3.0.3)

$$\begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}$$

we solve the regularised system

$$\begin{pmatrix} R + \tau I & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}$$

where I is the identity matrix and τ is the regularisation parameter. One can recover the solution $(\lambda\ \mu)^T$ to the original system (3.0.3) using iterated Tikhonov regularisation (attributed to Riley, 1955) by iterating for $k = 0, 1, 2, \ldots$

$$\begin{pmatrix} R+\tau I & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} r_k \\ s_k \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix} - \begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda_k \\ \mu_k \end{pmatrix}$$
$$\begin{pmatrix} \lambda_{k+1} \\ \mu_{k+1} \end{pmatrix} = \begin{pmatrix} \lambda_k \\ \mu_k \end{pmatrix} + \begin{pmatrix} r_k \\ s_k \end{pmatrix}$$

with $(\lambda_0\ \mu_0)^T = 0$ until the residual $(r_k\ s_k)^T$ is sufficiently small. The regularisation parameter τ can be chosen in addition to the covariance function parameters θ as detailed in Section 3.3, although Riley (1955) suggests choosing τ to be around $10^{2-\varrho} - 10^{3-\varrho}$ where ϱ is the desired precision. Note that numerically Tikhonov regularisation has the same effect as assuming there is measurement error with variance τ in the objective function samples (see Section 3.5).

3.6.4 Solving the Interpolation System

So far we have assumed that in order to fit a surrogate $s(x)$ to an objective function one has to solve the entire interpolation system (3.0.3). However, in our optimization strategies from Sections 5.2 and 5.4 we only update the interpolant by adding one or more points to the set sampled so far. Thus there is no need to refactorise the entire matrix system every time provided the weight matrix W is kept fixed at each stage. In view of this, we briefly outline an update strategy for the interpolation system and refer the reader to Björkman and Holmström (2000) for details. Note that in practice the weights are often recalculated at each stage in which case the strategy is not applicable.

Suppose we have sampled our objective function f at the points $x_1, \ldots, x_N$. Recall the interpolation system (3.0.3):

$$\begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}$$

Let

$$P = QT = \begin{pmatrix} \hat{Q} & V \end{pmatrix} \begin{pmatrix} \hat{T} \\ 0 \end{pmatrix} \tag{3.6.2}$$

be the QR factorisation of P, where Q is orthogonal, T upper triangular and let

$$V^T R V = L L^T$$

be the Cholesky factorisation of $V^T R V$, where L is lower triangular. This is possible as the columns of V span the null space of P^T and so $\lambda = Vv$ for some v which by the conditional positive definiteness of the RBF (see Definition 3.2) gives

$$v^T V^T R V v > 0 \quad \forall\, v \in \mathbb{R}^{N-M} \setminus \{0\}$$

i.e. that $V^T R V$ is indeed positive definite. We now solve the interpolation system (3.0.3) as follows:

1. Solve $Lu = V^T y$ for u.
2. Solve $L^T v = u$ for v.
3. Find $\lambda = Vv$.
4. Solve $\hat{T}\mu = \hat{Q}^T(y - R\lambda)$ for μ.

Note that this procedure (including the factorisations) requires $O(N^3)$ operations. Now suppose we have sampled our objective function at an additional point x_{N+1} and we wish to update our surrogate. We can do this as follows:

1. Update the correlation matrix
$$R^* = \begin{pmatrix} R & r(x_{N+1}) \\ r^T(x_{N+1}) & \varphi(0) \end{pmatrix}$$
where $r(x) = (\varphi(\|x - x_1\|_W), \ldots, \varphi(\|x - x_N\|_W))^T$ as before.
2. Update the polynomial basis matrix
$$P^* = \begin{pmatrix} P \\ p^T(x_{N+1}) \end{pmatrix}$$
where $p(x) = (\pi_1(x), \ldots, \pi_M(x))^T$ as before.
3. Calculate the QR factorisation of $P^* = Q^*T^* = \begin{pmatrix} \hat{Q}^* & V^* \end{pmatrix} \begin{pmatrix} \hat{T}^* \\ 0 \end{pmatrix}$ by
$$T^* = H \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix} P^*$$
where H is a suitably chosen matrix of Givens rotations (see Section 4 of Gill, Golub, Murray and Saunders, 1974, for details).
4. Let q denote the last column of Q^* and calculate $\tilde{w} = R^*q$
5. Let W denote the last $N - n$ columns of Q^* and calculate
$$\begin{pmatrix} \tau \\ \upsilon \end{pmatrix} = W^T \tilde{w}$$
6. Solve $Ll = \tau$ for l and calculate $s = \sqrt{\upsilon - l^T l}$.
7. Find the Cholesky factorisation of $V^{*T} R^* V^* = L^* L^{*T}$ from
$$L^* = \begin{pmatrix} L & 0 \\ l^T & s \end{pmatrix}$$
8. Solve the linear system by the previous algorithm.

Note that this procedure now only requires $O(N^2)$ operations (see Björkman and Holmström, 2000, for more details on this and the proposed algorithm). Thus we have reduced the amount of work required in solving the interpolation system at each successive iteration from $O(N^3)$ to $O(N^2)$.

3.6.5 Optimal Point Sets

The problem of finding the optimal set of sample points $\{x_1, \ldots, x_N \in \mathcal{D}\}$ for a given objective function (optimal in the sense of minimising the L_∞ error) is the subject of recent research in the RBF literature (see De Marchi, Schaback and Wendland, 2005) which we will extend in this section. Following Iske (2000), we know that available error estimates for the surrogate $s(x)$ possess the form

$$|f(x) - s(x)| \leq \|f\|_{\mathcal{N}_\varphi(\mathcal{D})} P_\varphi(x) \tag{3.6.3}$$

(see Theorem 11.4 in Wendland, 2005) where the objective function f is assumed to be an element of the Native space $\mathcal{N}_\varphi(\mathcal{D})$ (see Definition 3.4) and $P_\varphi(x)$ is the power function given by

$$P_\varphi^2(x) = \varphi(0) - 2u^T(x)r(x) + u^T(x)Ru(x)$$

(see Definition 11.2 in Wendland, 2005). Here $u(x)$ is the vector of cardinal functions which for a fixed $x \in \mathcal{D}$ can be evaluated by solving

$$\begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} u(x) \\ v(x) \end{pmatrix} = \begin{pmatrix} r(x) \\ p(x) \end{pmatrix} \tag{3.6.4}$$

where we have introduced the vector of functions $v(x)$.

It is evident from the error estimate (3.6.3) that one approach to obtain an almost optimal point set for $f \in \mathcal{N}_\varphi(\mathcal{D})$ is to reduce $\max_{x\in\mathcal{D}} P_\varphi(x)$ by sequentially adding maximisers of $P_\varphi(x)$ to the current point set. This is the approach taken by De Marchi et al. (2005) in their greedy algorithm in Section 4 of the paper. We however take a slightly different approach. Note that the power function can be rewritten as

$$P_\varphi^2(x) = \varphi(0) - u^T(x)r(x) - v^T(x)p(x)$$

(see Iske, 2000) which when combined with the linear system (3.6.4) gives the familiar form

$$\begin{aligned} P_\varphi^2(x) &= \varphi(0) - \begin{pmatrix} r(x) \\ p(x) \end{pmatrix}^T \begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} r(x) \\ p(x) \end{pmatrix} \\ &= \frac{e^2(x)}{\sigma^2}. \end{aligned}$$

Thus maximising the power function is equivalent to maximising $e(x)$, the square root of the posterior variance (i.e. the standard deviation) in the stochastic interpretation of RBF interpolation from Sections 3.1 and 3.2. As a result we can use the algorithms we shall develop in Chapter 6 to sequentially maximise $e(x)$ to obtain a point set which is almost optimal for objective functions in the Native space. However, we still require a small number of initial points for constructing $e(x)$ to start the algorithm (see Chapter 4). In fact, this is precisely the maximum error approach discussed in depth in Section 5.2.

Chapter 4

Initial Designs

Before we can even consider how to improve the surrogate model given by our framework from Chapter 3, we need to construct the surrogate to begin with. For this we need initial sample points $x_1, \ldots, x_N \in \mathcal{D}$ along with objective function samples at these points. Recall that the theory from Section 3.6 tells us that the set of points has to be Π_d^n-unisolvent, however a scattered set of greater than $M = \dim(\Pi_d^n)$ points will almost always satisfy this criterion. The question then is, how do we determine how many and which points to take? While for our purposes it does not really matter how many points, within reason, one takes initially (as more points will be added to the surrogate as we iterate the optimization framework from Section 5.2) more points to begin with does yield a better initial global picture of the objective function. Taking too many however, leads one to waste sample points on regions of the objective function one is not interested in.

What is perhaps more important is which points to take, that is how the points should be distributed in the domain $\mathcal{D}$. For example, a sensible approach is to ensure they are space filling, i.e. spread out so as to avoiding leaving large unsampled regions in the domain. Many such methods for generating initial sample points, or initial designs as they are called, exist in the design and analysis of computer experiments (DACE) literature (see Santner et al., 2003) as well as the RBF literature. We give a brief overview of relevant methods as well as a performance comparison on two dimensional test problems. These are chosen because the performance of individual designs can be thoroughly analysed since it is possible to visualise the designs and test functions.

4.1 Pseudo-Random Numbers

Perhaps the most obvious approach is to start with N iid uniformly distributed random numbers on our domain $\mathcal{D}$. This however, turns out to be a bad idea as such a design

will not be very good at filling the available space in $\mathcal{D}$, resulting in regions of the objective function being undersampled. Figure 4.1 illustrates this behaviour using two random designs on the unit square generated by different algorithms. It is important to note that generating

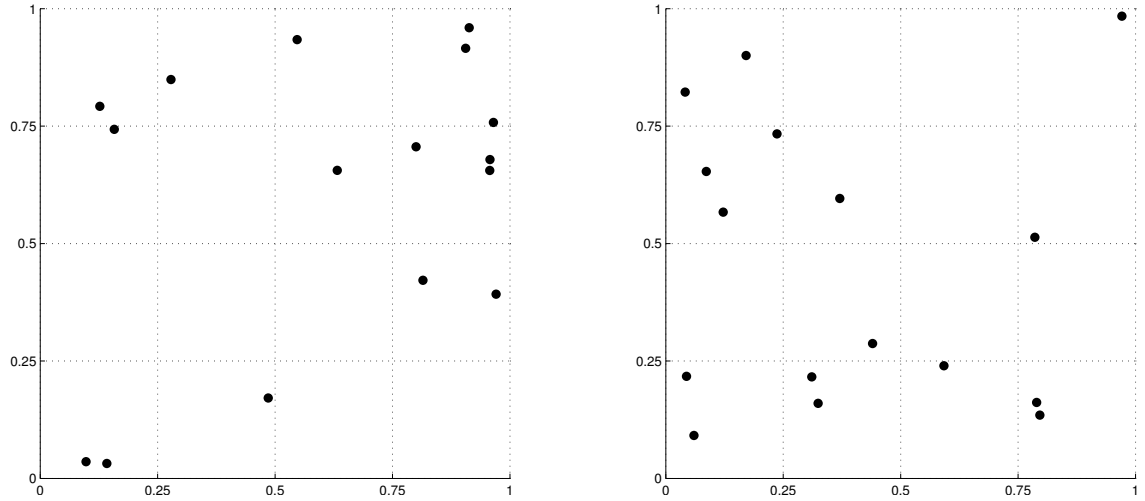

Figure 4.1: *Two random initial designs on the unit square for 16 points. These were generated using the Mersenne Twister algorithm (left) and ACORN generator (right). Notice how both designs leave large undersampled regions.*

iid uniformly distributed random variables is a far from easy problem since finite computers are inherently deterministic. As a result it is impossible to design an algorithm to generate truly random numbers, however one can get in some sense as close as one likes using so-called pseudo-random number generators. These are deterministic algorithms which generate sequences of numbers satisfying a number of randomness criteria. They are not considered to be truly random since they will always generate the same sequence of numbers given the same initial state. We refer the interested reader to Chapter 7 of Niederreiter (1992) for a comprehensive introduction to pseudo-random numbers.

One of the most popular and widely used algorithms for generating pseudo-random numbers is the Mersenne Twister algorithm, a twisted generalized feedback shift register generator. Unless otherwise stated, we use the $mt19937ar$ version of the Mersenne Twister algorithm from 2002 which has a period length of $2^{19937} - 1$ and is uniformly distributed in 623 dimensions (see Matsumoto and Nishimura, 1998, for more details).

Another algorithm one can use is the ACORN generator, a particular type of multiple recursive generator (Wikramaratna, 2008) which we will briefly describe. Given an integer modulus M, an integer seed $y_0^0 \in (0, M)$ such that y_0^0 and M are relatively prime and K initial integer values $y_0^1, \ldots, y_0^K \in [0, M)$ the K-th order ACORN generator generates N pseudo-random numbers $x_1^K, \ldots, x_N^K \in [0, 1)$ using the simple recursion

$$\begin{aligned} y_r^0 &= y_{r-1}^0 && r = 1, \ldots, N \\ y_r^p &= (y_r^{p-1} + y_{r-1}^p)_{\text{mod } M} && r = 1, \ldots, N;\ p = 1, \ldots, K \\ x_r^K &= y_r^K / M && r = 1, \ldots, N \end{aligned}$$

One can illustrate how the recursion proceeds in the following diagram from Wikramaratna (1989), where the arrows indicate which numbers are combined to calculate a particular value y_r^p.

$$\begin{array}{ccccccc} y_0^0 & \to & y_1^0 & \to & \cdots & \to & y_N^0 \\ & & \downarrow & & & & \downarrow \\ y_0^1 & \to & y_1^1 & \to & \cdots & \to & y_N^1 \\ & & \downarrow & & & & \downarrow \\ \vdots & & \vdots & & & & \vdots \\ & & \downarrow & & & & \downarrow \\ y_0^K & \to & y_1^K & \to & \cdots & \to & y_N^K \end{array}$$

The generator produces uniformly distributed random numbers in up to K dimensions and has a period length greater than M. Following the suggestions in the paper (Wikramaratna, 2008) we take $M = 2^{60}, K = 10$, $y_0^0 = 1$ and $y_0^p = 1$ for all $p = 1, \ldots, K$.

Most pseudo-random number generators will generate nearly uniform sample points $x_1, \ldots, x_N$ in the unit hypercube but we want these to lie in our domain $\mathcal{D}$. This is achieved by transforming each generated point x_i from the unit hypercube onto a general hyperrectangle $\prod_{k=1}^n [a_i, b_i]$, which is assumed to contain the domain $\mathcal{D}$, using the standard linear transformation $T\colon [0,1]^n \to \prod_{k=1}^n [a_i, b_i]$ defined by

$$T(x_i^k) = a_k + (b_k - a_k)x_i^k \tag{4.1.1}$$

for all $i = 1, \ldots, N$ and $k = 1, \ldots, n$. Here x_i^k denotes the k-th component of the i-th point x_i. If x_i lies outside the domain $\mathcal{D}$ it can simply be discarded until we obtain a point which does.

4.2 DACE Designs

The design and analysis of computer experiments (DACE) literature contains numerous examples of different strategies for initial designs, see for example Chapters 5 and 6 of Santner et al. (2003) and Section 5 of Koehler and Owen (1996) and references therein. In this section we will give a brief review of, in our opinion, the most interesting initial designs, starting with Latin hypercube designs. The idea behind Latin hypercube sampling, first proposed for computer experiments in McKay, Conover and Beckman (1979), is best described for the two dimensional case. Choose N sample points such that if we superimpose an $N \times N$ grid onto the unit square, there is exactly one point in each row and column. Formally, letting $[0,1]^n$ be the n-dimensional unit hypercube, *Latin hypercube* sample points $x_1, \ldots, x_N \in [0,1]^n$ are generated as follows: For each dimension $k = 1, \ldots, n$ let $\sigma_k(1), \ldots, \sigma_k(N)$ be an independent random permutation of $\{1, \ldots, N\}$. Also, for each $i = 1, \ldots, N$ let $\xi_{i,k}$ be an iid uniformly distributed random variable on $[0,1]$ independent of $\sigma_k(1), \ldots, \sigma_k(N)$. We then define x_i^k, the k-th component of x_i, as

$$x_i^k = h_k^{-1}\left(\frac{\sigma_k(i) - \xi_{i,k}}{N}\right)$$

where h_k is the cumulative distribution function (cdf) of x_k on some interval $[a, b]$. In general, this is taken to be the uniform distribution on $[0, 1]$ in which case we have $h_k \equiv id$, the identity function. The sample points are then mapped onto the domain $\mathcal{D}$ using (4.1.1).

Note that there is no guarantee that a Latin hypercube design will be space filling, so it is often desirable to impose a criterion to ensure this is the case. One approach is to impose a maximin or minimax criterion on the sample points (see Section 5.3 of Koehler and Owen, 1996), that is to say choose a Latin hypercube design $\mathcal{L} = \{x_1, \ldots, x_N\} \subset [0, 1]^n$ to either

$$\begin{aligned} \text{maximise} \quad & \min_{x,y \in \mathcal{L}} \|x - y\| && \text{(Maximin Criterion)} \\ \text{or minimise} \quad & \max_{x \in [0,1]^n} \min_{1 \le i \le N} \|x - x_i\| && \text{(Minimax Criterion).} \end{aligned}$$

In short, the maximin criterion ensures all design points are as far away from each other as possible and the minimax criterion ensures all points in the domain $\mathcal{D}$ are always close to a design point. Figure 4.2 illustrates the difference between a normal, a maximin and a minimax Latin hypercube design. The maximin and minimax designs are computed by generating a large set of Latin hypercube designs and choosing the designs which optimize the above criteria over this set.

A more general initial design can be constructed using orthognal arrays (Owen, 1992). Formally, an *orthogonal array* of strength t, denoted $\mathcal{OA}(N, M, q, t)$, is an $N \times M$ matrix A whose elements are taken from a set of q symbols (in general taken to be $\{0, 1, \ldots, q-1\}$) such that in any $N \times t$ submatrix each of the q^t possible rows occurs the same number of times (N/q^t). For example, the 4×3 matrix A given by

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

is $\mathcal{OA}(4, 3, 2, 2)$. To see this observe that the matrix only contains the 2 symbols $0, 1$ and in every 4×2 submatrix each of the 4 possible rows $0\ 0, 0\ 1, 1\ 0$ and $1\ 1$ occur once. Note that the definition imposes certain constraints on the possible values of N, M, q, t for which an orthogonal array exists (see Chapter 2 of Hedayat, Sloane and Stufken, 1999). A *randomised* orthogonal array is then an initial design with design points given by

$$x_i^k = \frac{\sigma_k(a_{i,k}) + \xi_{i,k}}{q}$$

where $a_{i,k}$ denotes the i, k-th component of A, $\sigma_k(0), \ldots, \sigma_k(q-1)$ is an independent random permutation of $\{0, \ldots, q-1\}$ and $\xi_{i,k}$ are again iid uniform random variables on $[0, 1]$. For example, the matrix $(x_1 \ \ldots \ x_N)$ of Latin hypercube samples with $\xi_{j,k} = 1/2$ for all j, k, called a Lattice Sample, is an orthogonal array of strength 1, that is $\mathcal{OA}(N, n, N, 1)$. It then follows that a randomised orthogonal array $\mathcal{OA}(N, n, N, 1)$ is a Latin hypercube design and thus one can use orthogonal arrays to generalise Latin hypercube and Lattice

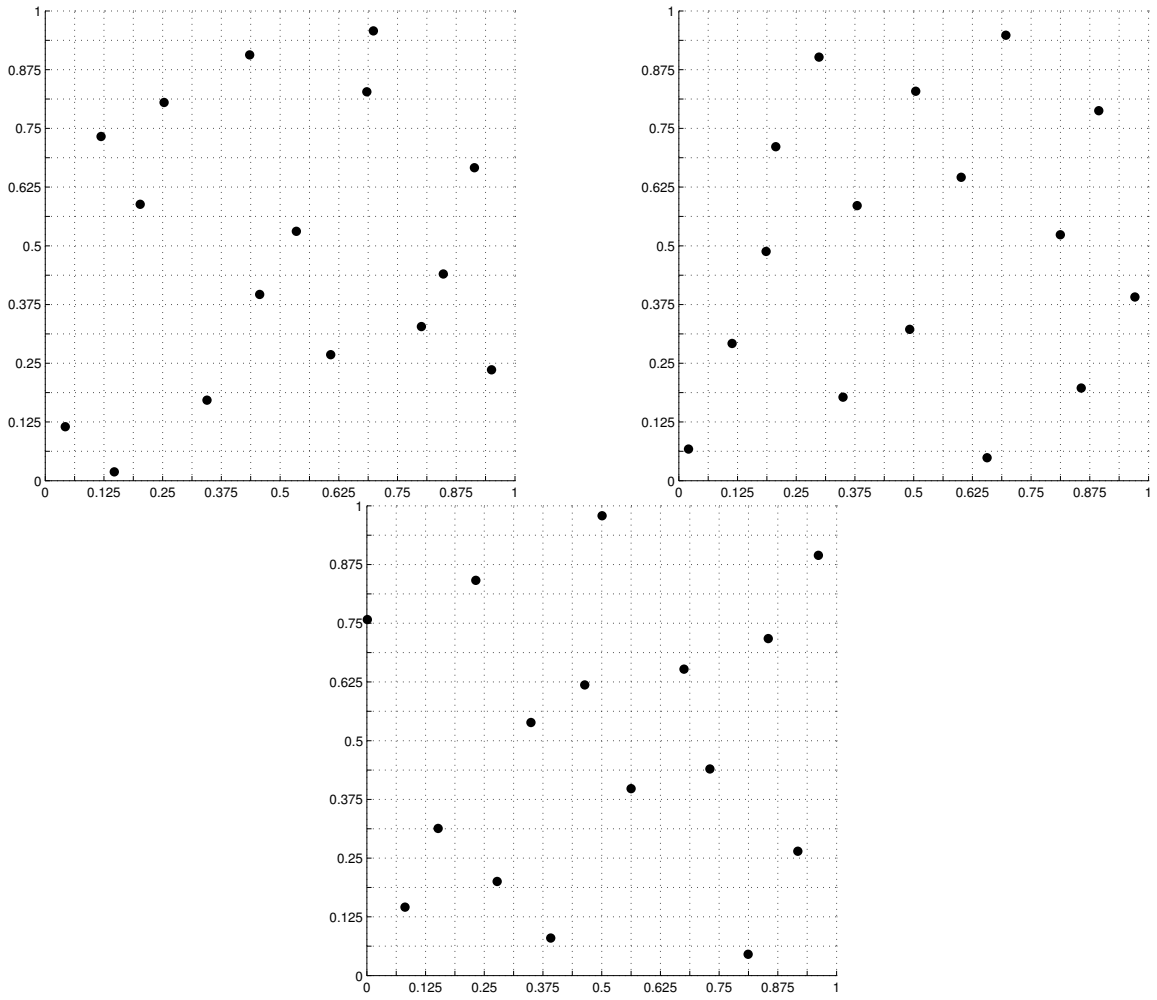

Figure 4.2: *A Latin hypercube (top left), maximin Latin hypercube (top right) and minimax Latin hypercube (bottom centre) design on the unit square for 16 points. Note that there is exactly one point in each row and column of the superimposed 16 × 16 grid, the defining property of Latin hypercubes. Also note how the maximin criterion is much better at filling the available space.*

Samples (see Owen, 1992). Moreover, one can even construct orthogonal array based Latin hypercube samples (see Tang, 1993). Again, it is generally a good idea to use a space-filling criterion where possible. Figure 4.3 overleaf illustrates a randomised orthogonal array and an orthogonal array based Latin hypercube.

Other initial design criteria make use of the fact that in the case of a RBF surrogate we have a probability distribution associated with our objective function. Namely, we assume that $f(x)$ is a realisation of a stochastic process F (see Sections 3.1 and 3.2 for more details on this). As an example, a *maximum entropy* design is one which maximises the entropy $E[-\ln \rho(F(\mathcal{L}))]$ over all designs $\mathcal{L} = \{x_1, \ldots, x_N\} \subset \mathcal{D}$ where $\rho(F(x))$ denotes the probability density function of $F(x)$. That is to say, a design $\mathcal{L}$ satisfying

$$\max_{\text{Designs } \mathcal{L}} E[-\ln \rho(F(\mathcal{L}))].$$

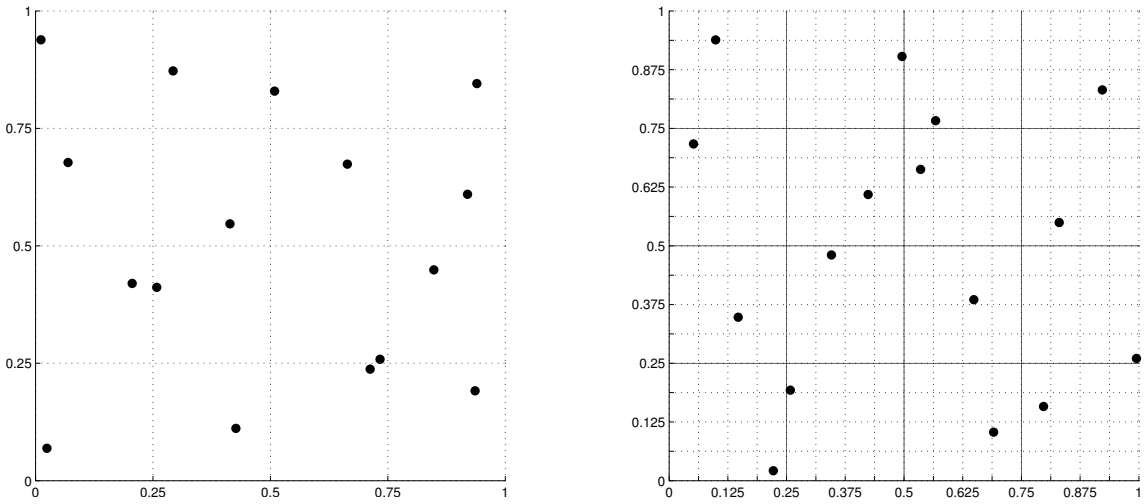

Figure 4.3: *A plot of the first two coordinates of a randomised* $\mathcal{OA}(16, 5, 4, 2)$ *(left) and a Latin hypercube based* $\mathcal{OA}(16, 5, 4, 2)$ *(right). Note that there is exactly one point in each of the large square regions. Moreover, in the case of the Latin hypercube based orthogonal array there is exactly one point in each row and column of the superimposed* 16×16 *grid.*

For a Gaussian stochastic process F it can be shown (see Section 6.2.1 of Santner et al., 2003) that a maximum entropy design maximises the determinant of the variance of the observed data y at the points $x_1, \ldots, x_N$ in the design. If we were to assume that the polynomial coefficients μ were known then according to (3.2.2) the maximum entropy design would be given by

$$\max_{x_1,\ldots,x_N \in \mathcal{D}} \det(\sigma^2 R).$$

However, in our derivation in Section 3.2 we placed a diffuse prior on μ and therefore we have according to (3.3.3) that the maximum entropy design is given by

$$\max_{x_1,\ldots,x_N \in \mathcal{D}} \det(\sigma^2 R) \det(P^T (\sigma^2 R)^{-1} P).$$

Figure 4.4 depicts maximum entropy designs for the Branin and Camel functions. A more sophisticated approach is taken by Busby, Farmer and Iske (2007) who develop a heirarchical adaptive gridding algorithm to generate a design using local maximum entropy designs, although this approach is better suited to applications where one is interested in accurately representing the entire objective function such as uncertainty analysis.

Another example is the *integrated mean squared error* design (see Section 6.2.2 of Santner et al., 2003) which minimises the integral of the mean squared error of prediction (3.1.1) over all designs $\mathcal{L} \subset \mathcal{D}$. That is to say, a design $\mathcal{L}$ satisfying

$$\min_{\mathcal{L} \subset \mathcal{D}} \int_{\mathcal{D}} \frac{1}{\sigma^2} E\Big[(s(x) - F(x))^2\Big] \, dx = \min_{\mathcal{L} \subset \mathcal{D}} \int_{\mathcal{D}} \frac{e^2(x)}{\sigma^2} dx$$

where $e^2(x)$ is the posterior variance from Sections 3.1 and 3.2. If on the other hand, one is interested in integrating the objective function $f(x)$ over $\mathcal{D}$ (as in Chapter 7) a natural

approach is to minimise the variance of the integral, given by (see O'Hagan, 1992)

$$\iint_{\mathcal{D}^2} \mathrm{Cov}\big[F(x), F(x')\big]\, dxdx' = \iint_{\mathcal{D}^2} e^2(x, x')dxdx'$$

where

$$e^2(x, x') = \sigma^2 \left[\varphi(0) - \begin{pmatrix} r(x) \\ p(x) \end{pmatrix}^T \begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} r(x') \\ p(x') \end{pmatrix}\right]$$

is the posterior covariance of the process F (see e.g. the derivation of $e^2(x)$ in Section 3.1). This criterion is briefly discussed in Chapter 7.

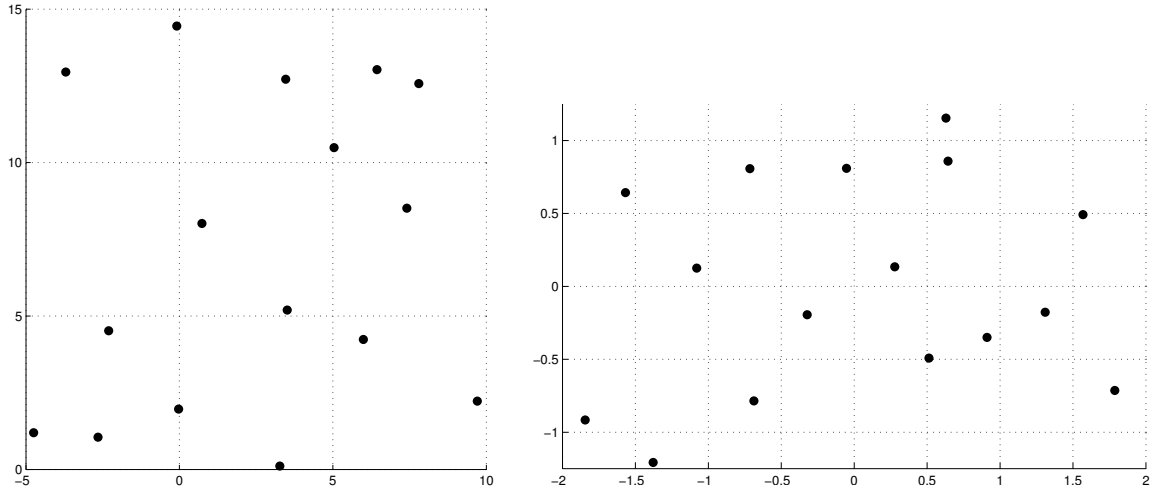

Figure 4.4: *16 point maximum entropy designs on* $[-5, 10] \times [0, 15]$ *for the Branin function (left) and on* $[-2, 2] \times [-1.25, 1.25]$ *for the Camel function (right).*

4.3 Low Discrepancy Sequences

Another commonly used type of initial design is the so-called low discrepancy sequence (sometimes also referred to as quasi-random sequence) which we will introduce in this section. We refer the interested reader to Niederreiter (1992) for more details on low discrepancy sequences especially in the context of Monte Carlo integration. As with Latin hypercube designs, low discrepancy sequences are defined on the unit hypercube and mapped onto more general domains $\mathcal{D}$ using (4.1.1). Firstly, let us introduce the concept of discrepancy which gives these sequences their name and the related but for our purposes more important concept of dispersion (cf. Definitions 2.2 and 6.2 in Niederreiter, 1992).

The *discrepancy* of a set of points $x_1, \ldots, x_N \in [0, 1]^n$ is given by

$$D(x_1, \ldots, x_N) = \sup_{\mathcal{G} \in \mathcal{J}} \left| \frac{S(\mathcal{G})}{N} - (b_1 - a_1)(b_2 - a_2) \ldots (b_n - a_n) \right|$$

where $\mathcal{J}$ is the family of all subintervals of $[0, 1]^n$ of the form $\mathcal{G} = \prod_{i=1}^n [a_i, b_i)$ and $S(\mathcal{G})$ is the number of points x_i for which $x_i \in \mathcal{G}$. One can think of the discrepancy of a set of points

$D(x_1, \ldots, x_N)$ as the biggest possible error when estimating the volume of the hyperrectangle $\mathcal{G}$ by sampling using the points $x_1, \ldots, x_N$ and using $S(\mathcal{G})/N$ as the estimate of its volume.

The *dispersion* of a set of points $x_1, \ldots, x_N \in [0,1]^n$ is given by

$$d(x_1, \ldots, x_N) = \max_{x \in [0,1]^n} \min_{1 \leq i \leq N} \|x - x_i\|$$

where $\|\cdot\|$ is some norm in $\mathbb{R}^n$. Thus a set of points $x_1, \ldots, x_N$ with low dispersion satisfies the minimax condition from Section 4.2 and thus has the property that any point in $[0,1]^n$ is always close to a design point x_i. We have the following theorem relating dispersion and discrepancy:

Theorem 4.1 (Niederreiter, 1992)**.** *For any set of points $x_1, \ldots, x_N \in [0,1]^n$ we have*

$$d(x_1, \ldots, x_N) \leq D(x_1, \ldots, x_N)^{1/n}$$

where the dispersion d is taken with respect to the ℓ_∞-norm.

Proof: See the proof of Theorem 6.6 in Niederreiter (1992). □

Note that as $\|\cdot\|_2 \leq \sqrt{n}\|\cdot\|_\infty$ the above is also true for the ℓ_2-norm, and more generally as all norms on a finite dimensional space are equivalent, any norm in $\mathbb{R}^n$. Thus every low discrepancy set of points is a low dispersion set of points, but not conversely. It therefore follows that an initial design $x_1, \ldots, x_N \in [0,1]^n$ with low discrepancy is equivalent to imposing a minimax condition on the initial design as defined in Section 4.2. Common examples of low discrepancy sequences are the Halton, Faure, Sobol and Niederreiter sequences (see Fox, 1986; Bratley, Fox and Niederreiter, 1992 and references therein). Figure 4.5 depicts the first 16 points of these sequences in two dimensions.

We will briefly describe how the Halton and Sobol sequences are constructed. Halton points (Halton, 1960) are created from van der Corput sequences as follows. Suppose $r \in \mathbb{N}_0$, the set of natural numbers with 0, p is prime and let

$$r = \sum_{i=0}^{K} a_i p^i \quad 0 \leq a_i < p,\ a_i \in \mathbb{N}_0$$

be the unique decomposition of r with respect to the prime base p. Define $h_p \colon \mathbb{N}_0 \to [0,1)$ via

$$h_p(r) = \sum_{i=0}^{K} \frac{a_i}{p^{i+1}}.$$

Then $h_{p,N} = \{h_p(r) : r = 0, 1, 2, \ldots, N\}$ is a van der Corput sequence. To generate $N+1$ Halton points in $[0,1)^n$, one takes n distinct primes and determines the corresponding van der Corput sequences $h_{p_1,N} \ldots, h_{p_n,N}$. Then one can form Halton points by taking the van der Corput sequences as coordinates giving

$$H_{n,N} = \{(h_{p_1}(r), \ldots, h_{p_n}(r)) : r = 0, 1, \ldots, N\},$$

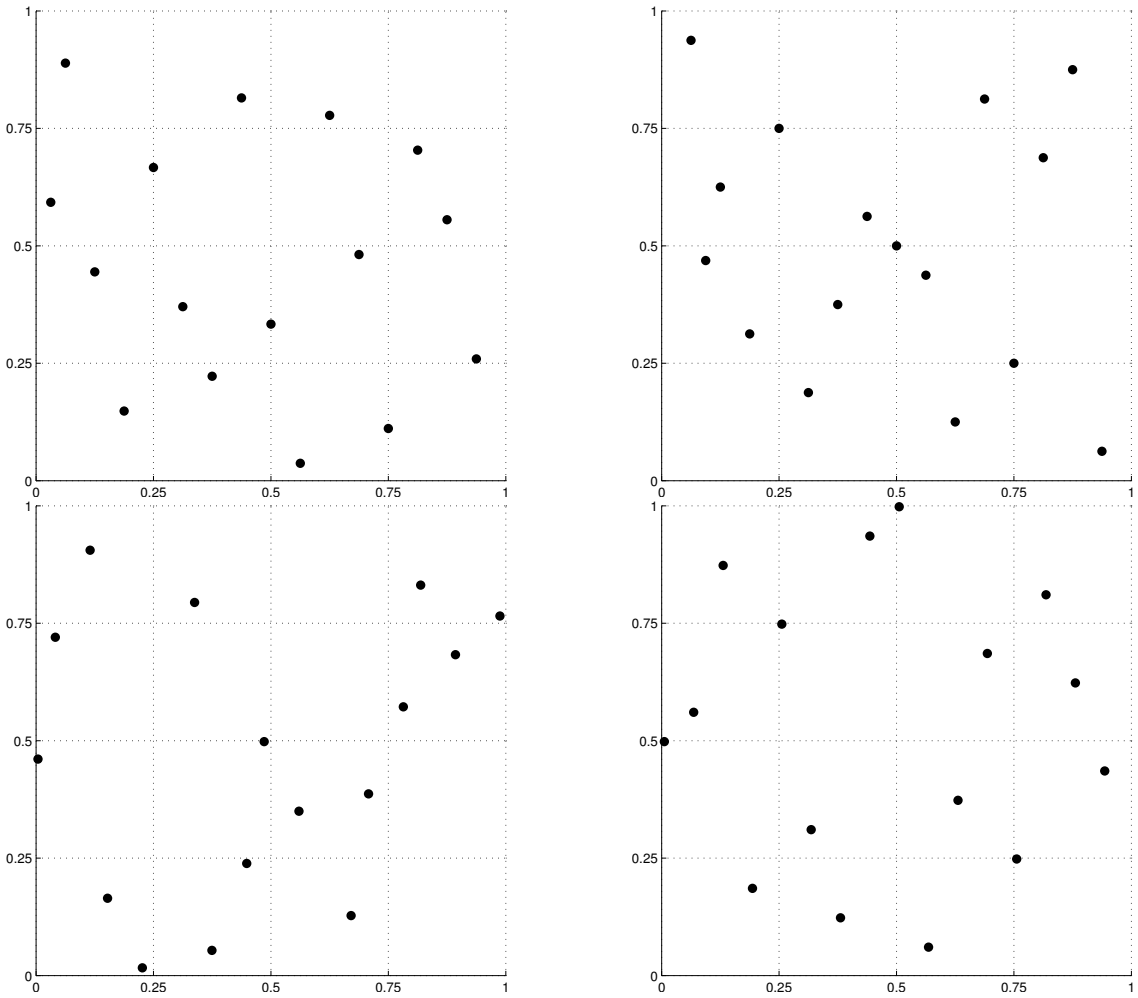

Figure 4.5: *A Halton (top left), Sobol (top right), Faure (bottom left) and Niederreiter (bottom right) design on the unit square for 16 points.*

a set of $N+1$ Halton points. Halton points are nested ($H_{S,M} \subset H_{S,N}$ for $M < N$) and can be constructed sequentially. In low dimensions the Halton sequence quickly fills up the unit cube in a well distributed pattern. Formally, it has a discrepancy of $O\left(\frac{\log^n N}{N}\right)$, see Theorem 3.6 in Niederreiter (1992).

Sobol sequences on the other hand are generated using an entirely different technique. To generate the j-th component of a Sobol sequence (see Bratley and Fox, 1988) we need to choose a primitive polynomial in $\mathbb{Z}_2 = \{0, 1\}$

$$P_j = x^{d_j} + a_1, jx^{d_j - 1} + \ldots + a_{d_j - 1, j}x + 1$$

where each coefficient $a_{i,j} \in \{0, 1\}$ and d_j is the polynomial degree. We then define the direction numbers $v_{i,j}$ in terms of odd integers $m_{i,j}$ as

$$v_{i,j} = \frac{m_{i,j}}{2^i}, \qquad m_{i,j} \text{ odd}, \, 0 < m_{i,j} < 2^i.$$

The integers $m_{i,j}$ are calculated from initial values $m_{1,j}, \ldots, m_{d_j,j}$ using the recurrence relation

$$m_{i,j} = 2a_{1,j}m_{i-1,j} \oplus 2^2 a_{2,j}m_{i-2,j} \oplus \ldots \oplus 2^{d_j-1}a_{d_j-1,j}m_{i-d_j+1,j} \oplus 2^{d_j}m_{i-d_j,j} \oplus m_{i-d_j,j}$$

where $\oplus$ denotes the binary exclusive 'or' operation (i.e. the integers $m_{i,j}$ are converted to binary and added). Then the j-th component of the i-th point in a Sobol sequence is given by

$$x_j^i = i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus \ldots$$

where i_k denotes the k-th binary digit of i. To get $O(\log^n N)$ discrepancy it suffices to choose any N different primitive polynomials.

4.4 Optimal Designs

A form of optimal design pursued by Iske (2000) is motivated by convergence and stability results for radial basis functions. Suppose we have a radial basis function surrogate $s(x)$ which interpolates the objective function at $x_1, \ldots, x_N \in \mathcal{D}$. Once again, we assume that the objective function $f \in \mathcal{N}_\varphi(\mathcal{D})$. Recall from Section 3.6 that the separation distance given by

$$q(x_1, \ldots, x_N) = \frac{1}{2} \min_{i \neq j} \|x_i - x_j\|$$

should not be too small so that solving the RBF interpolation system (3.0.3) is a stable process. Further, to ensure that the error in the interpolation is small, the fill distance given by

$$h(x_1, \ldots, x_N) = \max_{x \in \mathcal{D}} \min_{1 \leq i \leq N} \|x - x_i\|$$

should be small. Note that the fill distance h is the same as the dispersion of $\{x_1, \ldots, x_N\}$ and thus minimising h is the same as minimising the discrepancy (see Section 4.3). As it is not possible to simultaneously maximise q and minimise h, Iske suggests that one should compromise by maximising their ratio

$$p(x_1, \ldots, x_N) = \frac{q(x_1, \ldots, x_N)}{h(x_1, \ldots, x_N)}$$

which he defines as the uniformity of the design $\{x_1, \ldots, x_N\}$. Designs which minimise p are then referred to as optimally distributed. In particular for the unit square $[0, 1]^2$, Iske proves that the hexagonal grid is optimally distributed (see Figure 4.6). This suggests that perhaps in higher dimensions the lattices which give rise to the optimal kissing numbers (of which the hexagonal grid is an example) may be optimally distributed (see Subsection 6.5.4 for details on kissing numbers).

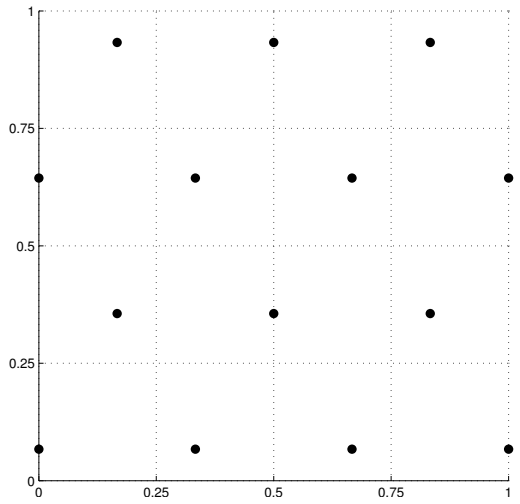

Figure 4.6: *An optimally distributed hexagonal grid design on the unit square for 14 points.*

4.5 Numerical Comparison

As is evident from this chapter, there are a large number of initial designs to choose from. It is far from obvious which ones, if any, are optimal for approximating a specific objective function. In order to address this, we shall try to give an indication of their performance in this section. We will consider two Dixon-Szegő test functions, the Branin and six hump camel back functions (Dixon and Szegő, 1978). Table 4.1 overleaf shows the discretised L_1, L_2 and L_∞ errors of a cubic spline RBF surrogate of the Branin function fitted to the different 16 point designs pictured in this chapter (with the exception of the optimally distributed design which has 14 points). Notice that the lowest L_1, L_2 and L_∞ errors are achieved by the Latin hypercube design. Table 4.2 overleaf shows the discretised errors of a cubic spline RBF surrogate of the camel function fitted to the different designs. This time the lowest L_1, L_2 and L_∞ errors are achieved by the space filling maximin Latin hypercube design. These results suggest that Latin hypercube based designs are best at minimising the L_1, L_2 and L_∞ errors as they tend to be very good at filling the available space in the domain. A more thorough comparison is necessary before drawing any general conclusions.

Initial Design	$\|s-f\|_1$	$\|s-f\|_2$	$\|s-f\|_\infty$
Mersenne Twister	6.007	8.586	42.229
Acorn Generator	15.007	22.907	89.889
Latin Hypercube	**4.167**	**5.989**	**29.839**
Maximin L. H.	4.354	8.629	56.325
Minimax L. H.	4.958	8.032	44.010
Orthogonal Array	9.113	13.095	53.870
L. H. Orthogonal Array	7.989	15.311	116.968
Entropy	9.650	13.056	52.491
Niederreiter	5.883	9.358	71.863
Halton	6.129	11.855	100.746
Sobol	7.895	15.207	128.551
Faure	6.754	11.690	72.941
Optimally Distributed	5.801	7.485	32.008

Table 4.1: *Accuracy of surrogate approximations using the different designs in the discretised L_1, L_2 and L_∞ norms to the Branin function on $[-5, 10] \times [0, 15]$. The smallest errors for each norm are denoted in bold.*

Initial Design	$\|s-f\|_1$	$\|s-f\|_2$	$\|s-f\|_\infty$
Mersenne Twister	0.785	1.063	5.480
Acorn Generator	1.021	1.389	3.872
Latin Hypercube	1.092	1.464	7.325
Maximin L. H.	**0.588**	**0.750**	**2.327**
Minimax L. H.	0.701	0.909	5.153
Orthogonal Array	0.859	1.294	7.157
L. H. Orthogonal Array	0.726	0.959	4.557
Entropy	1.061	1.443	7.460
Niederreiter	1.013	1.446	5.043
Halton	0.816	1.461	7.360
Sobol	0.797	1.141	4.906
Faure	0.698	0.926	2.920
Optimally Distributed	0.822	0.991	3.070

Table 4.2: *Accuracy of surrogate approximations using the different designs in the discretised L_1, L_2 and L_∞ norms to the camel function on $[-2, 2] \times [-1.25, 1.25]$. The smallest errors for each norm are denoted in bold.*

Chapter 5

Decision Theory

Once an initial surrogate for our expensive objective function is constructed, we are interested in further sampling the objective function so as to update and hopefully improve the surrogate model. In general this will be an iterative process, that is to say we keep using more points to build the surrogate until, ideally, the process as a whole converges to the global minimum of the objective function. It is therefore a predominant part of surrogate optimization algorithms and we will show that it often takes the form of minimising some expected loss function $l(x)$ at each iteration, although one can also use non-decision-theoretic cost functions within the resulting framework. This Bayesian decision theoretic viewpoint was first proposed by Mockus (1975). In particular, see Mockus (1989) as well as Chapter 6 of Törn and Žilinskas (1989) and references therein for the historical background. We will therefore start by introducing decision theory following the very accessible introduction given by Lindley (1985) and refer the interested reader to Berger (1985) for a comprehensive treatment of the underlying theory. It should be noted that decision theory usually deals with situations where one has to make a single decision. The theory of making a sequence of decisions can be formulated using dynamic programming and will be introduced in a subsequent section. A notable introduction to dynamic programming is given by its founder Bellman (2003) while an excellent in depth treatment can be found in Bertsekas (2005) and its companion volume.

5.1 Introduction to Decision Theory

Supposing you are faced with the everyday occurrence of having to make a decision d about something, whatever that may be. Let us denote the set of all possible decisions you could take under these circumstances by $\mathcal{D}$ (so that $d \in \mathcal{D}$). Associated with making these decisions will be uncertain events representing the uncertainty in the decision making process. Let $\mathcal{V}$

denote the set of all possible uncertain events relevant to the problem (so that $v \in \mathcal{V}$ is a particular uncertain event). As the events are uncertain you will have a probability distribution $\rho(v)$ on $\mathcal{V}$ and this will in general be subjective.

Now, with each possible decision and uncertain event you associate a loss $l(d, v)$ where $l : \mathcal{D} \times \mathcal{V} \to \mathcal{L} \subset \mathbb{R}$. The loss expresses your dissatisfaction at the outcome of taking decision d and event v subsequently occurring. The scale $\mathcal{L}$ of your loss is arbitrary, with the only constraint being that $\sup \mathcal{L}$ represents the worst possible outcome and $\inf \mathcal{L}$ the best possible outcome of the decision making process. The association should of course be consistent (in the sense that if you prefer l_1 to l_2 and l_2 to l_3 you must prefer l_1 to l_3). Furthermore, your sets $\mathcal{D}$ of decisions and $\mathcal{V}$ of uncertain events should be exclusive (that is you can only choose one decision d and only one uncertain event v subsequently occurs).

The fundamental theorem of Bayesian decision theory states that the natural approach is to take the decision which minimises ones expected loss (see Section 1.3.1 in Berger, 1985). The expected loss is given by the standard expression

$$l(d) = \int_{\mathcal{V}} l(d, v)\rho(v)dv \tag{5.1.1}$$

and hence the optimal decision is given by

$$d^* = \arg \inf_{d \in \mathcal{D}} l(d).$$

Let us clarify this with a discrete example: Suppose I am about to leave my house and am trying to decide whether or not to take my umbrella. The decision set clearly consists of two possibilities: The decision to take my umbrella d_U and the decision to not take my umbrella $d_{\neg U}$. Similarly, the set of uncertain events consists of the event of whether it will rain today v_R and whether it does not $v_{\neg R}$. Let us suppose I assign the uncertain events the probabilities $\rho(v_R) = 3/4$ and $\rho(v_{\neg R}) = 1/4$ because I think it tends to rain frequently where I live. Now I must assign losses to each combination of decision and subsequent uncertain event. Suppose I let $\mathcal{L} = [0, 1]$ and I assign losses as follows: $l(d_{\neg U}, v_R) = 1$ since I don't like getting wet; $l(d_{\neg U}, v_{\neg R}) = 0$ and $l(d_U, v_R) = 0$ since I stay dry and $l(d_U, v_{\neg R}) = 1/2$ since I have to carry the umbrella around for no reason. This can be summarised in a decision table

	v_R	$v_{\neg R}$
d_U	0	1/2
$d_{\neg U}$	1	0
$\rho(\cdot)$	3/4	1/4

Hence, the expected losses for d_U and $d_{\neg U}$ are given by

$$l(d_U) = l(d_U, v_R)\rho(v_R) + l(d_U, v_{\neg R})\rho(v_{\neg R}) = 0 \cdot 3/4 + 1/2 \cdot 1/4 = 1/8,$$
$$l(d_{\neg U}) = l(d_{\neg U}, v_R)\rho(v_R) + l(d_{\neg U}, v_{\neg R})\rho(v_{\neg R}) = 1 \cdot 3/4 + 0 \cdot 1/4 = 3/4$$

and so the optimal decision is to take the umbrella.

A somewhat more complicated decision problem arises when I observe data about the parameter v. That is, I have a prior distribution $\rho(v)$ for $\mathcal{V}$ which encompasses my prior beliefs (possibly based on previous experience of similar situations) about how probable the events v are. I then observe data x with associated likelihood $\rho(\mathsf{x}|v)$. The likelihood encompasses which values of v are more likely in view of the data. The subsequent posterior probability distribution $\rho(v|\mathsf{x})$ of v (that is the probability distribution of v after having observed the data) is given by Bayes' Theorem

$$\rho(v|\mathsf{x}) = \frac{\rho(v)\rho(\mathsf{x}|v)}{\int_{\mathcal{V}} \rho(v)\rho(\mathsf{x}|v)dv}.$$

We can then use the posterior probability for our decision problem, i.e. we use $\rho(v|\mathsf{x})$ instead of $\rho(v)$ when calculating the expected loss in (5.1.1).

Again, let us clarify by extending the above example: Suppose I take a look outside and observe that it is a clear sunny day. This information will of course influence the probability of rain today, hence denote it by w. Now suppose I assign the likelihoods as follows: $p(\mathsf{w}|v_R) = 1/10$ and $p(\mathsf{w}|v_{\neg R}) = 9/10$ because I think it is unlikely to rain on a sunny day. We then have

$$p(v_R)p(\mathsf{w}|v_R) = 3/4 \cdot 1/10 = 3/40,$$
$$p(v_{\neg R})p(\mathsf{w}|v_{\neg R}) = 1/4 \cdot 9/10 = 9/40$$

and so the updated (normalised) probabilities are $p(v_R|\mathsf{w}) = 1/4$ and $p(v_{\neg R}|\mathsf{w}) = 3/4$. Hence, the updated expected losses are

$$l(d_U) = l(d_U, v_R)p(v_R|\mathsf{w}) + l(d_U, v_{\neg R})p(v_{\neg R}|\mathsf{w}) = 0 \cdot 1/4 + 1/2 \cdot 3/4 = 3/8,$$
$$l(d_{\neg U}) = l(d_{\neg U}, v_R)p(v_R|\mathsf{w}) + l(d_{\neg U}, v_{\neg R})p(v_{\neg R}|\mathsf{w}) = 1 \cdot 1/4 + 0 \cdot 3/4 = 1/4$$

and so the optimal decision is to leave the umbrella at home.

5.2 One-stage Lookahead

It is natural to apply decision theory to the problem of deciding where to next sample our expensive objective function. This approach is often termed *one-stage lookahead* in the literature since we are concerned with making one decision at a time as opposed to the general case of multiple decisions, termed *multi-stage lookahead*, which we will consider in Section 5.4.

Recall from Sections 3.1 and 3.2 that we can view our objective function f as a realisation of a Gaussian stochastic process F conditioned on the data y with mean given by the surrogate $s(x)$ and variance given by the mean squared error $e^2(x)$. In view of this, it is natural to formulate a loss function $l(x, F(x))$ which expresses the dissatisfaction at event $F(x) \in \mathbb{R}$ occurring having made a decision to sample at $x \in \mathcal{D}$. Thus for our decision problem of where to next sample the objective function, the expected loss (5.1.1) becomes

$$l(x) = \int_{-\infty}^{\infty} l(x, v)\rho(v|y)dv \tag{5.2.1}$$

where $v = F(x)$ and the probability density function $\rho(v|y)$ is given by

$$\rho(v|y) = \frac{1}{e(x)}\phi\left(\frac{v - s(x)}{e(x)}\right) \tag{5.2.2}$$

as the stochastic process F conditioned on the data is Gaussian. Here $\phi(\cdot)$ denotes the pdf of the standard normal distribution. The optimal (one-stage) decision is then given by

$$x^* = \arg\inf_{x \in \mathcal{D}} l(x).$$

This motivates the approach of iteratively improving the surrogate approximation $s(x)$ using the minimiser of the expected loss function at each stage. The complete extended framework, using a general cost function (which will often be an expected loss function), is as follows:

One-stage Lookahead

0. Initialisation:

a) Set $t = 0$.

b) Let $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be a continuous objective function on some domain $\mathcal{D}$.

c) Let $\mathcal{X}_0 \subset \mathcal{D}$ denote an initial design, i.e. an initial set of points where f has been sampled (see Chapter 4).

d) Choose a suitable cost function $l(x)$ (see the remainder of this section).

1. Until a suitable stopping criterion is satisfied (see below), repeat the following procedure:

a) Construct a surrogate approximation $s_t(x)$ to the objective function f at $\mathcal{X}_t$ along with the corresponding error $e_t(x)$ (see Chapter 3).

b) Optimize the cost function $l_t(x)$ using a suitable branch and bound algorithm to obtain a minimiser x_t (see Chapter 6) and evaluate the objective function f at x_t.

c) Set $\mathcal{X}_{t+1} := \mathcal{X}_t \cup \{x_t\}$.

d) Set $t = t + 1$.

There are many possible choices of cost function in the literature (see Berger, 1985 and Sasena, 2002) and in this section we will describe and derive the most important ones for the problem at hand. We differentiate between two main classes of cost function, expected loss functions derived from (5.2.1) using the decision theoretic approach of Section 5.1 and cost functions motivated by heuristic reasoning which cannot be derived from (5.2.1) but can still be used in our framework. While the cost functions are presented as standalone strategies, there is no reason why one cannot implement an algorithm which makes use of a combination of these (such as the algorithm in Section 6.2.3 of Törn and Žilinskas, 1989, for example).

5.2.1 Expected Loss Functions

Surrogate Minimum

Perhaps the most obvious approach is to sample the objective function f at the global minimum of the surrogate $s(x)$, i.e. taking the loss function

$$l(x, v) = v$$

then (5.2.1) gives the expected loss as

$$l(x) = \int_{-\infty}^{\infty} v\rho(v|y)dv = s(x).$$

While this approach is simple and intuitive, it does in general run into problems. In particular, one cannot ensure convergence to a global minimum: If the resulting algorithm locates the basin of a local minimiser of f, it will in general sample within the basin until the local minimum is located (see Jones, 2001, for an example). In other words, this approach yields an algorithm which is predominantly local in its search for the global minimum of f. However, if our surrogate is an accurate representation of the underlying objective function, numerical results suggest this can be the fastest approach to finding the global minimum (see Section 8.1).

Probability of Improvement

Another approach (first proposed by Kushner, 1964, in the 1-D case) is to choose the next sample point to be the point at which the probability of improving the objective function $f(x)$ beyond some target value T is maximised, that is to maximise $\mathcal{P}(F(x) \leq T)$. Let the loss function be the step function

$$l(x, v) = \begin{cases} -1 & \text{if } v < T \\ 0 & \text{otherwise} \end{cases}$$

then (5.2.1) gives

$$l(x) = \int_{-\infty}^{T} -1 \cdot \rho(v|y)dv + \int_{T}^{\infty} 0 \cdot \rho(v|y)dv = -\Phi\left(\frac{T - s(x)}{e(x)}\right)$$

where Φ is the CDF of the standard normal distribution. Thus, we take the next sample at the minimiser of $l(x) = -\mathcal{P}(F(x) \leq T)$. Jones (2001) lets T be a percentage improvement κ (say, $\kappa = 0.25$) over the current smallest objective function value y_{min}, that is $T = y_{min} - \kappa|y_{min}|$.

Gutmann (2001) proposes an algorithm identical to the probability of improvement approach (as shown in depth by Žilinskas, 2010) which uses $T \in [-\infty, \min_{x\in\mathcal{D}} s(x)]$ where the choice $T = \min_{x\in\mathcal{D}} s(x)$ is only permitted if none of the function samples $x_i,\ i = 1, \ldots, N$ used to construct the surrogate $s(x)$ are global minimisers of $s(x)$. He proves that for surface spline basis functions and suitable choices of target values T the sequence of points generated by the algorithm for a continuous objective function f is dense (and therefore converges to

the global minimum by Theorem 2.1 or 2.2). Törn and Žilinskas (1989) also show that for suitable (albeit difficult to check) conditions on the correlation function $\varphi(\cdot)$, the probability of improvement approach with $T < \min_{x\in\mathcal{D}} s(x)$ converges to the global minimum.

Once again, an important consideration is choosing a suitable value of the parameter T which will in general depend on the choice of objective function. Taking too large a value (i.e. T close to $\min_{x\in\mathcal{D}} s(x)$) will cause the resulting algorithm to be predominantly local in its search for the global minimum whereas taking too small a value (i.e. T close to $-\infty$) will cause it to be excessively global (see Theorems 6.4 and 6.5 in Törn and Žilinskas, 1989). As with the lower confidence bound approach above, a possible solution is to cycle or try multiple values of T in an attempt to strike a balance between local and global searches. This is addressed by Holmström (2008) who proposes an extension of Gutmann's algorithm using multiple values of T which shows improved performance. Interestingly, Jones (2001) shows that taking multiple values of T is equivalent to taking multiple values of τ in the lower confidence bound approach.

Expected Improvement

An approach which has received much attention in the literature (first proposed by Mockus, Tiesis and Žilinskas, 1978) is to choose the next sample point to maximise the expected improvement over the current smallest objective function value $y_{min} := \min_{1\leq i\leq N} y_i$. In order to reformulate this as a minimisation problem, take the negative loss function to be the improvement that a new objective function evaluation at x achieves over the current smallest objective function value y_{min}

$$-l(x,v) = \max\{y_{min} - v, 0\} = \begin{cases} y_{min} - v & \text{if } v < y_{min} \\ 0 & \text{otherwise.} \end{cases}$$

The negative of the expected improvement is then given by (5.2.1)

$$\begin{aligned} -l(x) &= \int_{-\infty}^{\infty} \max\{y_{min} - v, 0\}\rho(v|y)dv \\ &= \int_{-\infty}^{y_{min}} (y_{min} - v)\rho(v|y)dv + \int_{y_{min}}^{\infty} 0 \cdot \rho(v|y)dv \\ &= y_{min}\int_{-\infty}^{y_{min}} \rho(v|y)dv - \int_{-\infty}^{y_{min}} v\rho(v|y)dv. \end{aligned}$$

Rewriting v in the second integral as $v - s(x) + s(x)$ and multiplying by $e(x)/e(x)$ gives

$$\begin{aligned} -l(x) &= y_{min}\int_{-\infty}^{y_{min}} \rho(v|y)dv - \int_{-\infty}^{y_{min}} (v - s(x) + s(x))\,\rho(v|y)dv \\ &= y_{min}\int_{-\infty}^{y_{min}} \rho(v|y)dv - e(x)\int_{-\infty}^{y_{min}} \frac{v - s(x)}{e(x)}\rho(v|y)dv - \int_{-\infty}^{y_{min}} s(x)\rho(v|y)dv \end{aligned}$$

which when $\rho(v|y)$ in the second integral is expanded according to (5.2.2) leads to

$$\begin{aligned}
-l(x) &= y_{min} \int_{-\infty}^{y_{min}} \rho(v|y)dv + e(x) \int_{-\infty}^{y_{min}} -\left(\frac{v - s(x)}{e^2(x)}\right) \phi\left(\frac{v - s(x)}{e(x)}\right) dv \\
&\quad - s(x) \int_{-\infty}^{y_{min}} \rho(v|y)dv \\
&= y_{min}\Phi\left(\frac{y_{min} - s(x)}{e(x)}\right) + e(x)\phi\left(\frac{y_{min} - s(x)}{e(x)}\right) - s(x)\Phi\left(\frac{y_{min} - s(x)}{e(x)}\right) \\
&= e(x)[z\Phi(z) + \phi(z)]
\end{aligned}$$

where $z = (y_{min} - s(x))/e(x)$. Thus the best choice of sample point $x \in \mathcal{D}$ is the one which minimises

$$l(x) = -e(x)[z\Phi(z) + \phi(z)]$$

which is precisely the expected improvement criterion. One can also use a generalised improvement criterion $-l(x, v) = \max\{(y_{min} - v)^k, 0\}$ as proposed in Schonlau (1997), where larger values of k lead to a more global search. Gramacy and Lee (2011) advocate the use of $\min_{x\in\mathcal{D}} s(x)$ in place of y_{min} in the expected improvement criterion.

What makes this approach stand out from the ones considered previously, is that there is no need (in the case of the standard expected improvement) to choose a value for an adjustable parameter and moreover, under certain assumptions convergence to the global minimum of f is guaranteed. In particular, Schonlau (1997) proves that if the number of possible sample points is finite, then the algorithm will visit them all and so will always find the global minimum. Locatelli (1997) proves that in the 1-D case when the stochastic process F is chosen to be a Wiener Process (Brownian Motion) the iterates generated by the algorithm will be dense and so (by Theorem 2.1 or 2.2) the algorithm will converge to the global minimum. More recently, Vazquez and Bect (2010) have proved that under mild assumptions on a positive definite correlation function $\varphi(\cdot)$ the iterates generated by the algorithm will be dense if the objective function is in the Native Space $\mathcal{N}_\varphi(\mathcal{D})$. Moreover, they have shown that for a continuous objective function f the iterates generated by the algorithm will be almost surely dense. However, the standard expected improvement algorithm is fairly local in its search for the global minimum and so can take a long time to locate it (see Section 8.1). The generalised version provides a possible solution to this problem, but once again it is a question of choosing a suitable value for the parameter k.

5.2.2 Heuristic Cost Functions

Maximum Error

As the standard deviation $e(x)$ of the conditioned process $F|y$ provides a measure of error in the surrogate fit, it makes sense to use this in determining where to further sample the objective function. Indeed, we have shown in Subsection 3.6.5 that $|f(x) - s(x)| \leq \|f\|_{\mathcal{N}_\varphi(\mathcal{D})} e(x)/\sigma$ for objective functions f in the Native Space $\mathcal{N}_\varphi(\mathcal{D})$ (recall that a Native Space is a RBF function space, see Definition 3.4). This motivates sampling the objective

function at the global maximum of the error $e(x)$, i.e. minimising $-e(x)$ and therefore taking the cost function

$$l(x) = -e(x).$$

As one would expect, this strategy leads to an algorithm which samples the surrogate in unexplored regions and so is excessively global in its search, in contrast to the surrogate minimum approach. Intuitively, one would expect such an algorithm to eventually sample a dense set (and thus converge to the global minimum of the objective function by Theorem 2.1 or 2.2) but this remains to be proved. A possible proof is as follows: Observe that when the maximiser x^* of the error $e(x)$ is used to construct the new surrogate, the corresponding new error evaluated at x^* is zero and so the maximum which was attained by $e(x)$ collapses in the new error. Under suitable assumptions, one would expect all maxima in the error to eventually collapse and thus for the algorithm to sample a dense set.

Lower Confidence Bound

A more sophisticated approach pursued by Cox and John (1997) is to take the next sample at the minimiser of a lower confidence bounding function, a compromise between sampling the surrogate minimum and the maximum error. Take the cost function

$$l(x) = s(x) - \tau\sigma e(x)$$

where σ^2 is as defined in Chapter 3 and τ is a suitably chosen positive constant. This results in an algorithm which searches for both global minima of the surrogate and in relatively unexplored regions. The emphasis of the latter is controlled by the parameter τ: As $\tau \to 0$ we recover the method of sampling the surrogate minimum (a local search strategy) and as $\tau \to \infty$ we sample only in unexplored regions, a very global search strategy. We propose a scaled version of the above cost function

$$l(x) = (1 - \tau)s(x) - \tau\sigma e(x)$$

which has the advantage that it varies between local search when $\tau \to 0$ and global search when $\tau \to 1$ (as opposed to ∞). The difficulty with the lower confidence bound approach is thus determining a suitable value of τ, which will in general depend on the choice of objective function. A possible solution is to vary τ or try several values of τ at each iteration of the algorithm so as to obtain a combination of local and global searches. Jones (2001) shows that such a method can resolve the sensitivity of the lower confidence bound approach to the parameter τ.

5.2.3 Other Strategies

As one would expect, not all strategies presented in the literature fit in to our general one-stage lookahead framework, that is to say they cannot be represented as the minimum of some relatively simple cost function. However, this does not prevent their use in our approach

provided one is able to perform the required global optimization. The branch and bound algorithms from Chapter 6 will in general not be applicable as the computation of suitable upper and lower bounds will not be possible. Nonetheless, we will mention some of the more interesting strategies as they are highly relevant to our problem.

A notable example proposed by Streltsov and Vakili (1999) takes into account the cost of evaluating the objective function. Let $c : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ be a function which measures the cost of evaluating the objective function $f(x)$ at x. We can then take the next sample of f in our one-stage lookahead framework at the solution to the equation

$$E[\max\{y_{min} - F(x), 0\}] = c(x)$$

i.e. the point x where the expected improvement of evaluating the objective function at x is offset by the cost of the evaluation.

Another interesting approach pursued by Gramacy and Lee (2011) is to extend the expected improvement criterion to the case of a noisy objective function with unknown constraint region $\mathcal{C}$. The idea here is to consider the expected improvement at a reference point $z \in \mathcal{D}$ given that the objective function has been (hypothetically) sampled at a candidate point $x \in \mathcal{D}$, i.e. the expected conditional improvement

$$eci(z|x) = E[\max\{y_{min} - F(z|x), 0\}]\,.$$

Putting a pdf $g(z)$ on z (for example $g(z)$ could be uniform for $z \in \mathcal{C}$ and zero otherwise) we can average out z to give the integrated expected conditional improvement

$$ieci(x) = -\int_{\mathcal{D}} eci(z|x)g(z)dz$$

and the next sample of the objective function is given by the maximiser of the integrated expected conditional improvement.

5.2.4 Stopping Criteria

Each of the above strategies for choosing where to further sample the objective function leads to an iterative global optimization algorithm, which in some cases is guaranteed to converge to the global minimum of the objective function in the limit. However, in order to be practical the algorithm must terminate in finite time and therefore we need a stopping criterion. This question has received little attention in the literature so we provide some suggestions and summarise existing criteria: Stop the one-stage lookahead when

- $\Delta y := |y_i - y_{i+1}| < \epsilon$
- $\Delta x := \|x_i - x_{i+1}\| < \epsilon$
- the expected improvement $E[\max\{y_{min} - F(x_i), 0\}] < \epsilon$
- the probability of improvement $\mathcal{P}(F(x_i) \leq T) < \epsilon$

- $y_{min} < \min l(x)$, where $l(x)$ is the lower confidence bounding cost function
- a finite number of iterations is exceeded

where $\epsilon > 0$ is a prescribed tolerance. For noisy objective functions, the natural ϵ to take is the error in the objective function. There is no general consensus in the literature as to which is the best approach, but generally one of the above is used.

5.3 Introduction to Dynamic Programming

Let us extend the theory of making decisions from Section 5.1 and suppose you are faced with having to make a sequence of decisions $d_1, \ldots, d_K$ each of which is associated with some uncertain event $v_1, \ldots, v_K$. As before, let $\mathcal{D}$ denote the set of all possible decisions so that $d_k \in \mathcal{D}$ and $\mathcal{V}$ the set of all possible uncertain events so that $v_k \in \mathcal{V}$ for all $k = 1, \ldots, K$. Suppose further that associated with the uncertain events there is a dynamical system whose state at any time k is given by z_k and which evolves according to

$$z_{k+1} = g(d_k, z_k, v_k)$$

where $g : \mathcal{D} \times \mathcal{Z}_k \times \mathcal{V} \to \mathcal{Z}_{k+1}$ is a known function. It is assumed that the system starts at some initial state $z_1 \in \mathcal{Z}_1$. In much the same way as before, you associate a loss $l_k(d_k, z_k, v_k)$ with each possible decision d_k, system state z_k and uncertain event v_k which expresses your dissatisfaction at the uncertain event v_k occurring having taken decision d_k. The optimal sequence of decisions $d_1^*, \ldots, d_K^*$ is then the one which minimises the total expected loss once all decisions have been made

$$E\left[\sum_{k=1}^{K} l_k(d_k, z_k, v_k)\right] \tag{5.3.1}$$

where the expectation is taken over the uncertain events $v_1, \ldots, v_K$. Minimising the expected loss (5.3.1) requires simultaneously optimizing over the decision variables $d_1, \ldots, d_K$ which is rather expensive. The key insight of dynamic programming is that the expected loss can be optimized sequentially, one decision at a time, using the dynamic programming recursion

$$\begin{aligned} C_K(z_K) &= \min_{d_K \in \mathcal{D}} E_{v_K}\Big[l_K(d_K, z_K, v_K)\Big|z_K\Big] \\ C_k(z_k) &= \min_{d_k \in \mathcal{D}} E_{v_k}\Big[l_k(d_k, z_k, v_k) + C_{k+1}\big(g(d_k, z_k, v_k)\big)\Big|z_k\Big] \end{aligned} \tag{5.3.2}$$

for $k = K-1, \ldots, 1$ where the optimal decisions $d_1^*, \ldots, d_K^*$ are the minimisers attained by the cost-to-go functions $C_1, \ldots, C_K$ (see Section 1.3 in Bertsekas, 2005). Here we view C_k as a function which assigns to each state z_k the total loss or cost $C_k(z_k)$ from time k to $K+1$ having made the optimal decisions $d_k^*, \ldots, d_K^*$ and for this reason is called a cost-to-go function. In particular the cost-to-go function $C_1(z_1)$ is equal to the minimum of the expected loss (5.3.1) obtained by the optimal decisions $d_1^*, \ldots, d_K^*$.

While the dynamic programming recursion is a very useful formulation, it is still computationally expensive as it is a series of nested global optimization problems. One notable

attempt to overcome this difficulty and simplify the dynamic programming recursion is the rollout algorithm (see Section 6.4 in Bertsekas, 2005). The idea behind rollout is to only consider one decision at a time with an approximate cost-to-go function. That is to say, for $k = 1, \ldots, K$ the k-th decision d_k is the minimiser of

$$E_{v_k}\Big[l_k(d_k, z_k, v_k) + \tilde{C}_{k+1}\big(g(d_k, z_k, v_k)\big)\big|z_k\Big]$$

where $\tilde{C}_{k+1}$ is an approximation to the true cost-to-go function C_{k+1} (for example, this could be another surrogate approximation). For the rollout algorithm $\tilde{C}_{k+1}$ is chosen to be the cost-to-go function of a given sequence of decisions $\tilde{d}_1, \ldots, \tilde{d}_K$ which are obtained heuristically (for example by an educated guess). One could equally calculate $\tilde{C}_{k+1}$ from the dynamic programming recursion (5.3.2) with an approximate $\tilde{C}_{k+2}$ leading to a multi-stage rollout algorithm.

5.4 Multi-stage Lookahead

The one-stage decision theoretic approach pursued in Section 5.2 is actually a simplification of the problem we would really like to solve: Supposing we can only afford to make K evaluations of our objective function, where should they be? It is natural to formulate this as a multi-stage decision problem, deciding where to take the next K points $x_1, \ldots, x_K \in \mathcal{D}$ at which to sample the objective function f. We present an extension to the approach set out in Betrò (1991) as proposed by Mockus (1975). Let the initial state z_1 be given by

$$z_1 = y$$

and the system equation be given by

$$z_{k+1} = g(x_k, z_k, F(x_k)) \quad \text{for } k = 1, \ldots, K$$

where

$$g(x, z, F(x)) = \begin{pmatrix} z \\ F(x) \end{pmatrix}$$

is the system evolution function. We then wish to minimise the expected loss after K steps using one of the loss functions given in Section 5.2 at each stage. For example, the improvement loss function for the multi-stage case would be

$$l(x, z, F(x)) = -\max\{\min(z) - F(x), 0\}$$

where $\min(z)$ denotes the smallest component of the vector z. We can then apply the dynamic programming recursion (5.3.2) to deduce that the cost-to-go functions are given by

$$\begin{aligned} C_K(z_K) &= \min_{x_K \in \mathcal{D}} E_{F(x_K)}\Big[l(x_K, z_K, F(x_K))\big|z_K\Big] \\ C_k(z_k) &= \min_{x_k \in \mathcal{D}} E_{F(x_k)}\Big[l(x_k, z_k, F(x_k)) + C_{k+1}\big(g(x_k, z_k, F(x_k))\big)\big|z_k\Big] \end{aligned}$$

for $k = K-1, \ldots, 1$ and the best choice of sample points are the minimisers $x_1, \ldots, x_K$ attained by the cost-to-go functions $C_1, \ldots, C_K$. While this is a very elegant formulation of the problem, the dynamic programming recursion is very hard to solve in practice as it is a sequence of nested global optimization problems involving expectations and thus integration. This is why simplifications are often introduced, such as assuming that the next decision is the last one by letting $K = 1$ which gives the one-stage lookahead framework we have discussed in depth in Section 5.2. Similarly, letting $K = 2$ gives the two-stage lookahead

$$
\begin{aligned}
C_2(z_2) &= \min_{x_2 \in \mathcal{D}} E_{F(x_2)}\Big[l(x_2, z_2, F(x_2))\Big|z_2\Big] \\
C_1(z_1) &= \min_{x_1 \in \mathcal{D}} E_{F(x_1)}\Big[l(x_1, z_1, F(x_1)) + C_2\big(g(x_1, z_1, F(x_1))\big)\Big|z_1\Big]
\end{aligned}
$$

which can be combined to give the nested optimization problem

$$
V_1(y) = \min_{x_1 \in \mathcal{D}} E_{F(x_1)}\left[l(x_1, y, F(x_1)) + \min_{x_2 \in \mathcal{D}} E_{F(x_2)}\left[l\left(x_2, \begin{pmatrix} y \\ F(x_1) \end{pmatrix}, F(x_2)\right) \,\middle|\, \begin{pmatrix} y \\ F(x_1) \end{pmatrix}\right] \,\middle|\, y\right].
$$

Note that while the inner expectation can be evaluated analytically for the various loss functions as in Section 5.2, the outer expectation in general cannot. As a result our optimization algorithms developed in Chapter 6 are not directly applicable. A possible simplification pursued in Osborne et al. (2009) is to sequentially sample $F(x_k)|z_k$ from the corresponding Gaussian distribution with mean $s(x_k)$ and variance $e^2(x_k)$ for $k = 1, \ldots, K$. This avoids having to do the expensive integration when calculating the expectation but it is not clear whether this is an effective approximation. A more sophisticated approach is to estimate the integrals sequentially using a Monte Carlo method as done by Gramacy and Lee (2011) for their integrated conditional expected improvement. As mentioned in Section 5.3, another possible simplification is the rollout algorithm. For other approaches to simplifying the dynamic programming recursion we refer the interested reader to Bertsekas (2005).

Chapter 6

Branch and Bound Algorithms

In the previous chapter we developed a framework based on decision theoretic reasoning for globally optimizing the expensive objective function $f(x)$. We have seen that as part of the solution to the general problem it is necessary to solve the ancillary global optimization problem

$$\min_{x \in \mathcal{D}} l(x)$$

where $l \colon \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ is a cost function from Section 5.2. In this chapter we will explore in detail solutions to this problem over various domains $\mathcal{D}$ using branch and bound algorithms. We have decided to use branch and bound algorithms for performing the necessary optimization as they have rigorous theoretical convergence guarantees and our use of radial basis function surrogates enables us to calculate bounds with relative ease.

6.1 Bound Constraints

Let us begin by describing a branch and bound algorithm for minimising the cost function $l(x)$ in the case of simple bound constraints on x. We will prove that the algorithm converges in a finite number of iterations to within a prescribed tolerance $\varepsilon > 0$ of the global minimum of $l(x)$.

6.1.1 Description of the Algorithm

We seek to optimize the cost function $l(x)$ from Section 5.2 over the domain $\mathcal{D}$ which in this section is an n-dimensional rectangle (or box). To this end we will use the canonical branch and bound algorithm (Horst, 1986) with bounds from Jones, Schonlau and Welch (1998) and Gutmann (2001). Before describing the algorithm in detail we first need some notation. Let $\mathcal{B} \subset \mathcal{D}$ denote an n-dimensional box and let $x_{\mathcal{B}}$ be the centre of $\mathcal{B}$ unless $\mathcal{B}$

has been locally searched, in which case it is the feasible point found by a constrained local search procedure (see step 1e in the algorithm below). Define $\alpha(\mathcal{B}), \beta(\mathcal{B})$ to be lower and upper bounds respectively on the global minimum of $l(x)$ within $\mathcal{B}$, i.e. $\alpha(\mathcal{B}), \beta(\mathcal{B})$ satisfy

$$\alpha(\mathcal{B}) \leq \min_{x \in \mathcal{B}} l(x) \leq \beta(\mathcal{B}).$$

Suitable choices for these bounds will be discussed below. We then follow the branch and bound algorithmic framework set out in Balakrishnan, Boyd and Balemi (1991):

Branch and Bound Algorithm

0. Initialisation:

a) Set $k = 0$ and $s = 0$.

b) Let $\mathcal{L}_0 = \{\mathcal{D}\}$ be the initial list of boxes.

c) Let $L_0 = \alpha(\mathcal{D})$ be the initial lower bound for $\min_{x \in \mathcal{D}} l(x)$.

d) Let $U_0 = \beta(\mathcal{D})$ be the initial upper bound for $\min_{x \in \mathcal{D}} l(x)$.

1. While $U_k - L_k > \varepsilon$, repeat the following procedure:

a) Remove from $\mathcal{L}_k$ boxes $\mathcal{B} \in \mathcal{L}_k$ such that $\alpha(\mathcal{B}) > U_k$.

b) Choose $\mathcal{B} \in \mathcal{L}_k$ such that $\alpha(\mathcal{B}) = L_k$.

c) Bisect $\mathcal{B}$ along its longest edge into $\mathcal{B}_I$ and $\mathcal{B}_{II}$. Set $\mathcal{L}_{k+1} := \mathcal{L}_k \cup \{\mathcal{B}_I, \mathcal{B}_{II}\}$ and remove $\mathcal{B}$ from $\mathcal{L}_{k+1}$.

d) If any boxes have been discarded in 1.a) set $s = 0$, otherwise set $s = s + 1$.

e) If $s > 2$ run an approximate constrained local search algorithm on all previously unsearched boxes $\mathcal{B}$ in $\mathcal{L}_{k+1}$, update $x_{\mathcal{B}}$ to be the minimiser found by the local search and set $s = 0$.

f) Set $L_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \alpha(\mathcal{B})$.

g) Set $U_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \beta(\mathcal{B})$.

h) Set $k = k + 1$.

2. Return U_k as the estimate of the global minimum of $l(x)$ over $\mathcal{D}$.

The idea behind the algorithm is to recursively partition the domain $\mathcal{D}$ into sub-boxes until a box of sufficiently small size containing the global minimum of $l(x)$ over $\mathcal{D}$ is found. Since it is possible to obtain bounds on the minimum of $l(x)$ over any box in $\mathcal{D}$, they can be used to discard boxes which cannot contain the global minimum, i.e. boxes whose lower bound is greater than the smallest upper bound. The algorithm is accelerated by running constrained local searches on suitable boxes to obtain more accurate upper bounds. This is achieved through the use of a heuristic from Pedamallu, Özdamar, Csendes and Vinkó (2008) which suggests running local searches on all previously unsearched boxes if no boxes are discarded after two successive iterations of the algorithm. We use a conjugate gradient based active set

method by Hager and Zhang (2006) for the local searches in our C++ code but in principle one can use any constrained local search algorithm. In practice the algorithm is stopped after a set amount of time and the solution is refined with a local solver.

We will now present suitable choices for the bounds α, β for the cost functions $l(x)$ given in Section 5.2. The cost function $l(x)$ can take many forms, but in all the sampling strategies proposed it is some combination of the RBF surrogate $s(x)$ and error $e(x)$. In fact, to obtain a lower (resp. upper) bound on $l(x)$ it suffices to obtain a lower (resp. upper) bound on $s(x)$ and an upper (resp. lower) bound on $e(x)$ as the cost functions are monotonic in $s(x)$ and $e(x)$. We will show monotonicity for the various cost functions later on in this section. Consider first the surrogate $s(x)$. Recall from Chapter 3 that $s(x)$ takes the form

$$s(x) = \sum_{k=1}^{M} \mu_k \pi_k(x) + \sum_{j=1}^{N} \lambda_j \varphi(\|x - x_j\|_W).$$

We bound each radial basis function term over a box $\mathcal{B} \subset \mathcal{D}$ using quadratic functions (as suggested in Gutmann, 2001 and Jones et al., 1998):

$$a_j + b_j \|x - x_j\|_W^2 \leq \varphi(\|x - x_j\|_W) \leq A_j + B_j \|x - x_j\|_W^2$$

for which details are given later in the convergence proof subsection. Then we can define a lower bounding function for the global minimum of the surrogate s over a box $\mathcal{B} \subset \mathcal{D}$ as

$$\alpha_s(\mathcal{B}) = \min_{x \in \mathcal{B}} \{p_s(x) + q_s(x)\} \tag{6.1.1}$$

where

$$p_s(x) := \sum_{k=1}^{M} \mu_k \pi_k(x)$$

$$q_s(x) := \sum_{\substack{j=1 \\ \lambda_j > 0}}^{N} \lambda_j (a_j + b_j \|x - x_j\|_W^2) + \sum_{\substack{j=1 \\ \lambda_j < 0}}^{N} \lambda_j (A_j + B_j \|x - x_j\|_W^2).$$

Note that in our implementation we use the cubic spline RBF where the polynomial term is linear and hence it is trivial to find its minimum over a box $\mathcal{B}$ since $\alpha_s(\mathcal{B})$ is then separable and piecewise quadratic. For other choices of RBF one can use interval arithmetic (Neumaier, 2004) or polynomial optimization (Lasserre, 2001) to find the minimum. We define the upper bounding function for the global minimum of the surrogate s over a box $\mathcal{B} \subset \mathcal{D}$ as

$$\beta_s(\mathcal{B}) = s(x_{\mathcal{B}}). \tag{6.1.2}$$

Consider now the error $e(x)$. Recall from Sections 3.1 and 3.2 that the error $e(x)$ is given by

$$e^2(x) = \sigma^2 \left[\varphi(0) - L(\xi)\right]$$

where

$$L(\xi) = \xi^T A^{-1} \xi, \quad A = \begin{pmatrix} R & P \\ P^T & 0 \end{pmatrix}, \quad \xi = (\, r(x)\; p(x)\,)^T.$$

The idea here is to underestimate $L(\xi)$ by a convex relaxation $C(\xi)$ for which details are given later in the convergence proof subsection. Following Jones et al. (1998) and Gutmann (2001), define an upper bounding function for the global maximum of the error $e(x)$ over a box $\mathcal{B}$ as

$$\beta_e(\mathcal{B}) = \max_{\xi \in [\underline{\xi}, \bar{\xi}]} \sigma(\varphi(0) - C(\xi))^{1/2}. \tag{6.1.3}$$

Similarly to before, a suitable lower bounding function for the global maximum of the error $e(x)$ over a box $\mathcal{B} \subset \mathcal{D}$ is

$$\alpha_e(\mathcal{B}) = e(x_{\mathcal{B}}). \tag{6.1.4}$$

We will now show that the cost functions from Section 5.2, are monotonic in $s(x)$ and $e(x)$. Clearly this must be the case for the surrogate minimum and maximum error cost functions as these only depend on $s(x)$ and $-e(x)$ respectively. The lower confidence bound cost function is linear in $s(x)$ and $-e(x)$ so monotonicity follows trivially. Consider the probability of improvement cost function $l(x)$, then

$$\frac{\partial(-l(x))}{\partial s(x)} = -\frac{1}{e(x)}\phi\left(\frac{T - s(x)}{e(x)}\right) < 0 \tag{6.1.5}$$

and

$$\frac{\partial(-l(x))}{\partial e(x)} = -\frac{(T - s(x))}{e^2(x)}\phi\left(\frac{T - s(x)}{e(x)}\right) > 0 \tag{6.1.6}$$

provided $T \leq \min_{x \in \mathcal{D}} s(x)$. Thus the cost function is monotonic in $s(x)$ and $e(x)$ under suitable assumptions on T. For the expected improvement cost function $l(x)$, Jones et al. (1998) have shown that

$$\frac{\partial(-l(x))}{\partial s(x)} = -\Phi\left(\frac{y_{min} - s(x)}{e(x)}\right) < 0 \tag{6.1.7}$$

and

$$\frac{\partial(-l(x))}{\partial e(x)} = \phi\left(\frac{y_{min} - s(x)}{e(x)}\right) > 0 \tag{6.1.8}$$

and hence that the cost function is monotonic in $s(x)$ and $e(x)$.

We can now define the lower bounding function for the surrogate minimum approach as

$$\alpha(\mathcal{B}) = \alpha_s(\mathcal{B}) \tag{6.1.9}$$

for the maximum error approach as

$$\alpha(\mathcal{B}) = -\beta_e(\mathcal{B}) \tag{6.1.10}$$

for the lower confidence bound approach as

$$\alpha(\mathcal{B}) = \alpha_s(\mathcal{B}) - \nu\sigma\beta_e(\mathcal{B}), \tag{6.1.11}$$

for the probability of improvement approach as

$$\alpha(\mathcal{B}) = -\Phi\left(\frac{T - \alpha_s(\mathcal{B})}{\beta_e(\mathcal{B})}\right) \tag{6.1.12}$$

(provided $T \leq \min_{x \in \mathcal{D}} s(x)$) and for the expected improvement approach as

$$\alpha(\mathcal{B}) = -\beta_e(\mathcal{B}) \left[\zeta(\mathcal{B})\Phi(\zeta(\mathcal{B})) + \phi(\zeta(\mathcal{B}))\right] \tag{6.1.13}$$

where

$$\zeta(\mathcal{B}) = \frac{y_{min} - \alpha_s(\mathcal{B})}{\beta_e(\mathcal{B})}.$$

The upper bounding functions $\beta(\mathcal{B})$ are exactly the same expressions with α and β interchanged.

6.1.2 Proof of Convergence

Consider the following two conditions on the upper and lower bounds α and β over a compact set $\mathcal{C} \subset \mathcal{D}$:

(C1) $\alpha(\mathcal{C}) \leq \min_{x \in \mathcal{C}} l(x) \leq \beta(\mathcal{C})$

(C2) $\forall \varepsilon > 0 \; \exists \delta > 0$ s.t. $\forall \mathcal{C} \subset \mathcal{D}, \; \text{size}(\mathcal{C}) \leq \delta \implies \beta(\mathcal{C}) - \alpha(\mathcal{C}) \leq \varepsilon$

where size($\mathcal{C}$) is some measure of the size of $\mathcal{C}$, which will be defined for specific sets $\mathcal{C}$ later. Essentially, the first condition states that α and β should be bounds on the minimum of $l(x)$ over $\mathcal{C}$ and the second condition that the bounds become more accurate as $\mathcal{C}$ gets smaller. Let us consider the specific case when the compact set $\mathcal{C}$ is a box $\mathcal{B}$, and define size($\mathcal{B}$) to be the maximum half-length of the sides of $\mathcal{B}$. We then have from the Appendix in Balakrishnan et al. (1991) that the above branch and bound algorithm computes the global minimum of the cost function $l(x)$ to within an absolute accuracy $\varepsilon > 0$ in a finite number of iterations if α and β satisfy the conditions (C1) and (C2) above. Recall from the previous section that we can obtain lower (resp. upper) bounds on the various cost functions in terms of lower bounds α_s (resp. upper bounds β_s) on $s(x)$ and upper bounds β_e (resp. lower bounds α_e) on $e(x)$ as the utility functions are monotonic in $s(x)$ and $e(x)$. In order to prove convergence of the branch and bound algorithm for these cost functions, it suffices to show that α_s, β_s and α_e, β_e satisfy similar conditions to (C1), (C2) as it then easily follows that α, β satisfy conditions (C1), (C2) above.

Before we prove convergence of the algorithm, we briefly digress to discuss how the quadratic bounds on the radial basis function are chosen (as described in Gutmann, 2001 and Jones et al., 1998). Without loss of generality, assume that the radial basis function $\varphi(\cdot)$ is convex (as it will either be convex or concave). Define $r_j := \|x - x_j\|_W^2$ and note that it is trivial to find

$$l_j := \min_{x \in \mathcal{B}} r_j$$

and

$$u_j := \max_{x \in \mathcal{B}} r_j$$

so that l_j, u_j are exact bounds on r_j over $\mathcal{B}$. This is because u_j is always attained at one of the vertices of the box $\mathcal{B}$ and l_j is attained at x_j if x_j is contained in $\mathcal{B}$ or at the closest point in $\mathcal{B}$ to x_j. The above bounds lead to the following lemma:

Lemma 6.1. *Let l_j, u_j be defined as above. Then for $j = 1, \ldots, N$ we have*

$$\forall \varepsilon > 0 \; \exists \delta_j > 0 \; \textit{s.t.} \; \forall \mathcal{B} \subset \mathcal{D} \; \forall x \in \mathcal{B}, \; \textit{size}(\mathcal{B}) \leq \delta_j$$
$$\Longrightarrow \begin{cases} \varphi(\|x - x_j\|_w) - \varphi(\sqrt{l_j}) \leq \varepsilon \\ \varphi(\sqrt{u_j}) - \varphi(\|x - x_j\|_w) \leq \varepsilon \end{cases}$$

Proof: Follows from the above bounding procedure and the uniform continuity and monotonicity of $\varphi(\cdot)$. Note that as $\mathcal{D}$ is compact and $\varphi(\cdot)$ is continuous it follows from the Heine-Cantor Theorem (see Munkres, 1997, Theorem 27.6, page 176) that $\varphi(\cdot)$ is uniformly continuous. □

We then wish to bound our radial basis function using quadratics as follows

$$a_j + b_j r_j \leq \varphi(\sqrt{r_j}) \leq A_j + B_j r_j$$

and since $\varphi(\sqrt{r_j})$ is convex, we can take $A_j + B_j r_j$ to be the chord from l_j to u_j and $a_j + b_j r_j$ as the tangent to $\varphi(\sqrt{r_j})$ at $\frac{1}{2}(l_j + u_j)$ (see Figure 6.1 below). For example, for our choice

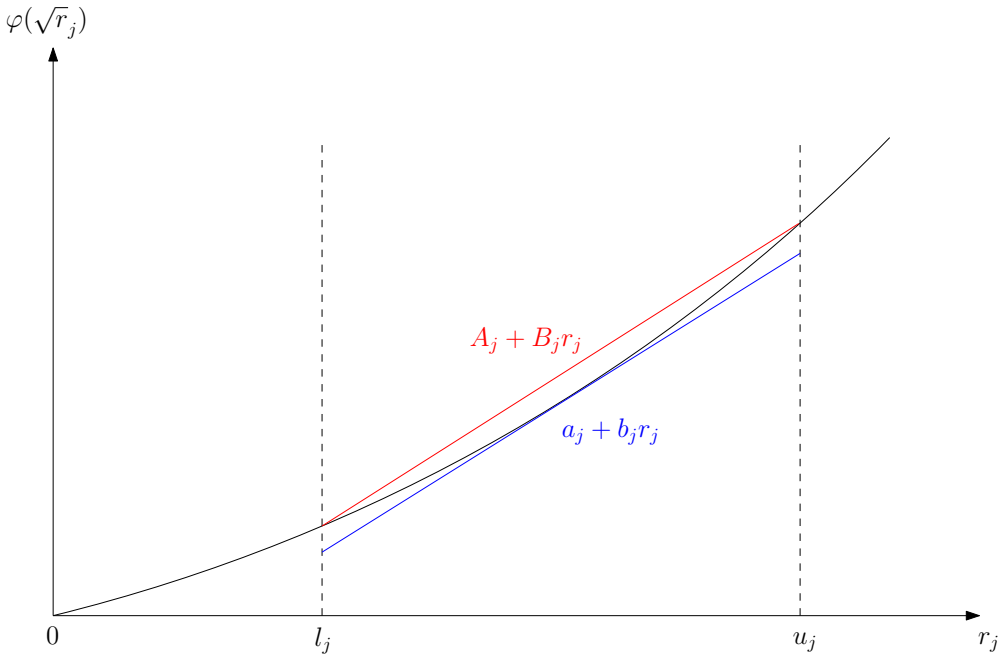

Figure 6.1: *Lower and upper bounding functions for $\varphi(\sqrt{r_j})$.*

of cubic RBF $\varphi(\sqrt{r_j}) = r_j^{3/2}$ and we want $A_j + B_j r_j = \varphi(\sqrt{r_j})$ at l_j and u_j. Hence A_j, B_j satisfy the equations $A_j + B_j l_j = l_j^{3/2}$ and $A_j + B_j u_j = u_j^{3/2}$ which one can solve to obtain

$$A_j = \frac{-l_j u_j}{\sqrt{l_j} + \sqrt{u_j}}, \qquad B_j = \frac{1}{\sqrt{l_j} + \sqrt{u_j}} + \sqrt{u_j}.$$

Also, we want $a_j + b_j r_j = \varphi(\sqrt{r_j})$ at $\frac{1}{2}(l_j + u_j)$ with $b_j = \varphi'(\sqrt{r_j})$. Solving these equations, one obtains

$$a_j = -\frac{(l_j + u_j)^{3/2}}{4\sqrt{2}}, \qquad b_j = \frac{3\sqrt{l_j + u_j}}{2\sqrt{2}}.$$

It is clear that as size($\mathcal{B}$) $\to 0$ we have that $u_j - l_j \to 0$ and thus the quadratic upper and lower bounds converge uniformly to $\varphi(\sqrt{r}_j)$. We formalise this observation in the following lemma:

Lemma 6.2. *Let each radial basis function $\varphi(\|x - x_j\|_W)$ be bounded as above. Then for $j = 1, \ldots, N$ we have*

$$\forall \varepsilon > 0 \; \exists \delta_j > 0 \; \textit{s.t.} \; \forall \mathcal{B} \subset \mathcal{D} \; \forall x, y \in \mathcal{B}, \; \textit{size}(\mathcal{B}) \leq \delta_j$$
$$\implies \begin{cases} \varphi(\|x - x_j\|_W) - (a_j + b_j \|y - x_j\|_W{}^2) \leq \varepsilon \\ A_j + B_j \|y - x_j\|_W{}^2 - \varphi(\|x - x_j\|_W) \leq \varepsilon \end{cases}$$

Proof: Follows from the above bounding procedure and the uniform continuity and convexity of $\varphi(\cdot)$. □

This immediately leads to

Lemma 6.3. *Define*

$$r_s(x) := \sum_{j=1}^{N} \lambda_j \varphi(\|x - x_j\|_W)$$

Then with the above bounding procedure the following result holds:

$$\forall \varepsilon > 0 \; \exists \delta > 0 \; \textit{s.t.} \; \forall \mathcal{B} \subset \mathcal{D} \; \forall x, y \in \mathcal{B}, \; \textit{size}(\mathcal{B}) \leq \delta \implies r_s(x) - q_s(y) \leq \varepsilon$$

Proof: Let $\varepsilon > 0$ be arbitrary. We have from Lemma 6.2 that for $j = 1, \ldots, N$ for which $\lambda_j \neq 0$, $\exists \delta_j > 0$ s.t. $\forall \mathcal{B} \subset \mathcal{D} \; \forall x, y \in \mathcal{B}$

$$\text{size}(\mathcal{B}) \leq \delta_j \implies \begin{cases} \varphi(\|x - x_j\|_W) - (a_j + b_j \|y - x_j\|_W{}^2) \leq \varepsilon / N\lambda_j & \text{for } \lambda_j > 0 \\ A_j + B_j \|y - x_j\|_W{}^2 - \varphi(\|x - x_j\|_W) \leq \varepsilon / N(-\lambda_j) & \text{for } \lambda_j < 0 \end{cases}$$

So take $\delta := \min_j \delta_j$. We then have that $\forall \mathcal{B} \subset \mathcal{D} \; \forall x, y \in \mathcal{B}$

$$\begin{aligned}
&\text{size}(\mathcal{B}) \leq \delta \implies r_s(x) - q_s(y) \\
&= \sum_{j=1}^{N} \lambda_j \varphi(\|x - x_j\|_W) - \sum_{\substack{j=1 \\ \lambda_j > 0}}^{N} \lambda_j (a_j + b_j \|y - x_j\|_W{}^2) - \sum_{\substack{j=1 \\ \lambda_j < 0}}^{N} \lambda_j (A_j + B_j \|y - x_j\|_W{}^2) \\
&= \sum_{\substack{j=1 \\ \lambda_j > 0}}^{N} \lambda_j [\varphi(\|x - x_j\|_W) - (a_j + b_j \|y - x_j\|_W{}^2)] + \sum_{\substack{j=1 \\ \lambda_j < 0}}^{N} (-\lambda_j) [(A_j + B_j \|y - x_j\|_W{}^2) - \varphi(\|x - x_j\|_W)] \\
&\leq \sum_{\substack{j=1 \\ \lambda_j > 0}}^{N} \lambda_j [\varepsilon / N\lambda_j] + \sum_{\substack{j=1 \\ \lambda_j < 0}}^{N} (-\lambda_j) [\varepsilon / N(-\lambda_j)] = \varepsilon
\end{aligned}$$

□

To prove convergence, we also need an additional lemma:

Lemma 6.4. *Let $p \in \Pi_d^n$ be a polynomial on a compact set $\mathcal{S} \subset \mathbb{R}^n$. We then have that*

$$\forall \varepsilon > 0\ \exists \delta > 0\ \textit{s.t.}\ \forall \mathcal{B} \subset \mathcal{S}\ \forall x, y \in \mathcal{B},\ \textit{size}(\mathcal{B}) \leq \delta \implies |p(x) - p(y)| \leq \varepsilon$$

Proof: Follows trivially from the uniform continuity of $p(\cdot)$. Note that as $\mathcal{S}$ is compact and $p : \mathcal{S} \to \mathbb{R}$ continuous (as it is a polynomial) we have by the Heine-Cantor Theorem (see Munkres, 1997, Theorem 27.6, page 176) that it is uniformly continuous. □

Now we are in a position to prove the following theorem.

Theorem 6.5. *The bounds α_s and β_s given above in* (6.1.1) *and* (6.1.2) *respectively satisfy the following two conditions:*

(C1s) $\alpha_s(\mathcal{B}) \leq \min_{x \in \mathcal{B}} s(x) \leq \beta_s(\mathcal{B})$

(C2s) $\forall \varepsilon > 0\ \exists \delta > 0\ \textit{s.t.}\ \forall \mathcal{B} \subset \mathcal{D},\ \textit{size}(\mathcal{B}) \leq \delta \implies \beta_s(\mathcal{B}) - \alpha_s(\mathcal{B}) \leq \varepsilon$

Proof: (C1s) Clearly, the value of $s(\cdot)$ at any point in $\mathcal{B}$ is an upper bound on the minimum of $s(\cdot)$ over $\mathcal{B}$. Also, by construction $p_s(x) + q_s(x)$ underestimates $s(x)$ over $\mathcal{B}$ and thus $\alpha_s(\mathcal{B}) = \min_{x \in \mathcal{B}}\{p_s(x) + q_s(x)\}$ underestimates $\min_{x \in \mathcal{B}} s(x)$.

(C2s) Let $\varepsilon > 0$ be arbitrary. For clarity of exposition define for all $\mathcal{B} \subset \mathcal{D}$,

$$x^*_{\mathcal{B}} := \arg\min_{x \in \mathcal{B}}\{p_s(x) + q_s(x)\}$$

and let c denote some point in $\mathcal{B}$ (not necessarily the midpoint). We have from Lemma 6.3 that there exists $\delta_1 > 0$ such that for any $\mathcal{B} \subset \mathcal{D}$

$$\text{size}(\mathcal{B}) \leq \delta_1 \implies r_s(c) - q_s(x^*_{\mathcal{B}}) \leq \varepsilon/2$$

We also have from Lemma 6.4 that there exists $\delta_2 > 0$ such that for any $\mathcal{B} \subset \mathcal{D}$

$$\text{size}(\mathcal{B}) \leq \delta_2 \implies p_s(c) - p_s(x^*_{\mathcal{B}}) \leq \varepsilon/2$$

So take $\delta := \min\{\delta_1, \delta_2\}$. We then have that for any $\mathcal{B} \subset \mathcal{D}$

$$\begin{aligned} \text{size}(\mathcal{B}) \leq \delta \implies \beta_s(\mathcal{B}) - \alpha_s(\mathcal{B}) &= p_s(c) - p_s(x^*_{\mathcal{B}}) + r_s(c) - q_s(x^*_{\mathcal{B}}) \\ &\leq \varepsilon/2 + \varepsilon/2 = \varepsilon. \end{aligned}$$

□

We now establish bounds on the error $e(x)$ and start by describing how to underestimate the quadratic form $L(\xi) = \xi^T A^{-1} \xi$ by a convex relaxation $C(\xi)$ over an interval $[\underline{\xi}, \overline{\xi}]$. Consider the convex relaxation

$$C(\xi) := L(\xi) + \gamma(\xi - \underline{\xi})^T(\xi - \overline{\xi})$$

where $\gamma := \max\{0, -\lambda_{min}\}+\epsilon$ controls the extent of the relaxation, λ_{min} denotes the smallest eigenvalue of A^{-1}, ϵ is a small positive constant and

$$\underline{\xi}_j = \begin{cases} \varphi(\sqrt{l_j}) \text{ for } j = 1, \ldots, N \\ \min_{x \in \mathcal{B}} \pi_{j-n}(x) \text{ for } j = N+1, \ldots, N+M \end{cases}$$

$$\overline{\xi}_j = \begin{cases} \varphi(\sqrt{u_j}) \text{ for } j = 1, \ldots, N \\ \max_{x \in \mathcal{B}} \pi_{j-n}(x) \text{ for } j = N+1, \ldots, N+M. \end{cases}$$

determine the interval over which the convex relaxation applies. Recall that for our implementation using a cubic RBF the polynomial basis $\{\pi_k\}_{k=1}^M$ is linear and so it is trivial to find the minimum or maximum of each π_k over a box $\mathcal{B}$. For other choices of basis function one can use interval arithmetic (Neumaier, 2004) or polynomial optimization (Lasserre, 2001) as before. Crucially $C(\xi)$ is convex as its Hessian, given by $2\left(A^{-1} + \gamma I\right)$, is strictly positive definite (by virtue of choosing a small $\epsilon > 0$) and it underestimates $L(\xi)$ over $[\underline{\xi}, \overline{\xi}]$ since for any $\xi \in [\underline{\xi}, \overline{\xi}]$, $\gamma(\xi - \underline{\xi})^T(\xi - \overline{\xi}) \leq 0$. Thus we have relaxed the problem to a convex quadratic programming problem which can be solved efficiently. We use QuadProg++, an implementation of the algorithm by Goldfarb and Idnani (1983) in our C++ code. To prove convergence, we also need the following lemma:

Lemma 6.6. *We have that*

$$\forall \varepsilon > 0 \; \exists \delta > 0 \; s.t. \; \forall \mathcal{B} \subset \mathcal{D} \; \forall \zeta, \xi \in [\underline{\xi}, \overline{\xi}], \; size(\mathcal{B}) \leq \delta \implies L(\zeta) - C(\xi) \leq \varepsilon$$

Proof: Let $\varepsilon > 0$ be arbitrary. For clarity of exposition define $\forall \mathcal{B} \subset \mathcal{D}$

$$x_k^l := \arg\min_{x \in \mathcal{B}} \pi_k(x)$$

$$x_k^u := \arg\max_{x \in \mathcal{B}} \pi_k(x)$$

We have from Lemma 6.1 that for $j = 1, \ldots, N \; \exists \delta_j > 0$ s.t.

$$\forall \mathcal{B} \subset \mathcal{D} \; \forall x \in \mathcal{B}, \; \text{size}(\mathcal{B}) \leq \delta_j \implies \begin{cases} \varphi(\|x - x_j\|_W) - \varphi(\sqrt{l_j}) \leq \sqrt{\varepsilon / 2\gamma(N+M)} \\ \varphi(\sqrt{u_j}) - \varphi(\|x - x_j\|_W) \leq \sqrt{\varepsilon / 2\gamma(N+M)} \end{cases}$$

and from Lemma 6.4 (as $\pi_k(x)$ is a polynomial) that for $k = 1, \ldots, M \; \exists \delta_k > 0$ s.t.

$$\forall \mathcal{B} \subset \mathcal{D} \; \forall x, y \in \mathcal{B}, \; \text{size}(\mathcal{B}) \leq \delta_k \implies |\pi_k(x) - \pi_k(y)| \leq \sqrt{\varepsilon / 2\gamma(N+M)}$$

As $\xi^T A^{-1} \xi$ is a quadratic form and thus a polynomial in ξ we have from Lemma 6.4 that $\exists \delta^* > 0$ s.t.

$$\forall \mathcal{B} \subset \mathcal{D} \; \forall \zeta, \xi \in [\underline{\xi}, \overline{\xi}], \; \text{size}(\mathcal{B}) \leq \delta^* \implies |\zeta^T A^{-1} \zeta - \xi^T A^{-1} \xi| \leq \varepsilon / 2$$

So take $\delta := \min_{j,k}\{\delta_j, \delta_k, \delta^*\}$. We then have that $\forall \mathcal{B} \subset \mathcal{D} \; \forall \zeta, \xi \in [\underline{\xi}, \overline{\xi}]$

$$\text{size}(\mathcal{B}) \leq \delta \implies$$

$$
\begin{aligned}
|L(\xi) - C(\zeta)| &= |\zeta^T A^{-1}\zeta - (\xi^T A^{-1}\xi + \gamma(\xi - \underline{\xi})^T(\xi - \overline{\xi}))| \\
&\leq |\zeta^T A^{-1}\zeta - \xi^T A^{-1}\xi| + \gamma(\xi - \underline{\xi})^T(\overline{\xi} - \xi) \\
&\leq \varepsilon/2 + \gamma \begin{pmatrix} \varphi(\|x - x_1\|_W) - \varphi(\sqrt{l_1}) \\ \vdots \\ \varphi(\|x - x_N\|_W) - \varphi(\sqrt{l_N}) \\ \pi_1(x) - \pi_1(x_1^l) \\ \vdots \\ \pi_M(x) - \pi_M(x_M^l) \end{pmatrix}^T \begin{pmatrix} \varphi(\sqrt{u_1}) - \varphi(\|x - x_1\|_W) \\ \vdots \\ \varphi(\sqrt{u_N}) - \varphi(\|x - x_N\|_W) \\ \pi_1(x_1^u) - \pi_1(x) \\ \vdots \\ \pi_M(x_M^u) - \pi_M(x) \end{pmatrix} \\
&= \varepsilon/2 + \gamma([\varphi(\|x - x_1\|_W) - \varphi(\sqrt{l_1})][\varphi(\sqrt{u_1}) - \varphi(\|x - x_1\|_W)] \\
&\quad + \ldots + [\varphi(\|x - x_N\|_W) - \varphi(\sqrt{l_N})][\varphi(\sqrt{u_N}) - \varphi(\|x - x_N\|_W)] \\
&\quad + [\pi_1(x) - \pi_1(x_1^l)][\pi_1(x_1^u) - \pi_1(x)] \\
&\quad + \ldots + [\pi_M(x) - \pi_M(x_M^l)][\pi_M(x_M^u) - \pi_M(x)]) \\
&\leq \varepsilon/2 + \gamma\left(\frac{N\varepsilon}{2\gamma(N+M)} + \frac{M\varepsilon}{2\gamma(N+M)}\right) = \varepsilon
\end{aligned}
$$

□

Now we are in a position to prove the following theorem:

Theorem 6.7. *The bounds α_e and β_e given above in* (6.1.4) *and* (6.1.3) *respectively satisfy the following two conditions:*

(C1e) $\alpha_e(\mathcal{B}) \leq \max_{x\in\mathcal{B}} e(x) \leq \beta_e(\mathcal{B})$

(C2e) $\forall\varepsilon > 0\ \exists\delta > 0$ *s.t.* $\forall\mathcal{B} \subset \mathcal{D},\ size(\mathcal{B}) \leq \delta \implies \beta_e(\mathcal{B}) - \alpha_e(\mathcal{B}) \leq \varepsilon$

Proof: (C1e) Clearly, the value of $e(\cdot)$ at any point in $\mathcal{B}$ is a lower bound on the maximum of $e(\cdot)$ over $\mathcal{B}$. Also, by construction $C(\xi)$ underestimates $L(\xi)$ over $\mathcal{B}$ and thus $\beta_e(\mathcal{B}) = \max_{\xi\in[\underline{\xi},\overline{\xi}]} \sigma(\varphi(0) - C(\xi))^{1/2}$ overestimates $\max_{x\in\mathcal{B}} e(x)$.

(C2e) Let $\varepsilon > 0$ be arbitrary. For clarity of exposition define $\forall\mathcal{B} \subset \mathcal{D}$,

$$\xi^* := \arg\max_{\xi\in[\underline{\xi},\overline{\xi}]} \sigma(\varphi(0) - C(\xi))^{1/2}$$

and let ξ^c denote some point in $[\underline{\xi}, \overline{\xi}]$. From the uniform continuity of the square root function we have that

$$\exists\,\varepsilon^* > 0 \text{ s.t. } (\beta_e)^2 - (\alpha_e)^2 \leq \varepsilon^* \implies \beta_e - \alpha_e \leq \varepsilon.$$

We have from Lemma 6.6, that $\exists\delta > 0$ s.t. $\forall\mathcal{B} \subset \mathcal{D}$

$$\text{size}(\mathcal{B}) \leq \delta \implies L(\xi^c) - C(\xi^*) \leq \varepsilon^*.$$

We then have that $\forall\mathcal{B} \subset \mathcal{D}$

$$
\begin{aligned}
\text{size}(\mathcal{B}) \leq \delta \implies (\beta_e)^2 - (\alpha_e)^2 &= \varphi(0) - L(\xi^c) - \varphi(0) + C(\xi^*) \\
&= L(\xi^c) - C(\xi^*) \leq \varepsilon^*
\end{aligned}
$$

and hence $\forall \mathcal{B} \subset \mathcal{D}$

$$\text{size}(\mathcal{B}) \leq \delta \implies \beta_e - \alpha_e \leq \varepsilon \qquad \square$$

Finally, we obtain the main convergence result of this section:

Theorem 6.8. *The bounds α and β given above in (6.1.9) – (6.1.13) satisfy conditions (C1) and (C2). Hence, by the proof in Balakrishnan et al. (1991) the proposed algorithm converges in a finite number of iterations with an absolute accuracy $\varepsilon > 0$.*

Proof: (C1) We know from the previous section that the cost functions from Section 5.2 are monotonic in $s(x)$ and $e(x)$. In particular, they decrease as $s(x)$ decreases and $e(x)$ increases (and vice versa). Therefore, it follows trivially from conditions (C1s), (C1e) of Theorems 6.5, 6.7 that the bounds α, β given in (6.1.9) – (6.1.13) satisfy condition (C1).

(C2) For the surrogate minimum, maximum error and lower confidence bound cost functions it follows trivially from conditions (C2s), (C2e) of Theorems 6.5, 6.7 that the respective bounds α, β given in (6.1.9) – (6.1.11) satisfy condition (C2). For the probability of improvement cost function, we give the following proof: Let $\varepsilon > 0$ be arbitrary. First of all note that the probability of improvement cost function can be viewed as a function of $s(x)$ and $e(x)$. We then have from the monotonicity (6.1.5), (6.1.6) that it has bounded gradient for our bounds on $s(x)$ and $e(x)$ over $\mathcal{B}$ since $\mathcal{D}$ is compact. In particular, this means it is uniformly continuous and there exists $\delta^* > 0$ such that for any $\mathcal{B} \subset \mathcal{D}$

$$\|(\beta_s(\mathcal{B}), \alpha_e(\mathcal{B})) - (\alpha_s(\mathcal{B}), \beta_e(\mathcal{B}))\|_2 \leq \delta^* \implies l(\beta_s(\mathcal{B}), \alpha_e(\mathcal{B})) - l(\alpha_s(\mathcal{B}), \beta_e(\mathcal{B})) \leq \varepsilon$$

where $l(\cdot, \cdot)$ denotes the probability of improvement cost function as a function of $s(x)$ and $e(x)$. We have from conditions (C2s) and (C2e) of Theorems 6.5, 6.7 that there exist $\delta_1, \delta_2 > 0$ such that for any $\mathcal{B} \subset \mathcal{D}$

$$\begin{aligned} \text{size}(\mathcal{B}) \leq \delta_1 &\implies \beta_s(\mathcal{B}) - \alpha_s(\mathcal{B}) \leq \delta^*/\sqrt{2}, \\ \text{size}(\mathcal{B}) \leq \delta_2 &\implies \beta_e(\mathcal{B}) - \alpha_e(\mathcal{B}) \leq \delta^*/\sqrt{2}. \end{aligned}$$

So take $\delta := \min\{\delta_1, \delta_2\}$. Then for any $\mathcal{B} \subset \mathcal{D}$, $\text{size}(\mathcal{B}) \leq \delta$ implies

$$\begin{aligned} \|(\beta_s(\mathcal{B}), \alpha_e(\mathcal{B})) - (\alpha_s(\mathcal{B}), \beta_e(\mathcal{B}))\|_2 &= \left((\beta_s(\mathcal{B}) - \alpha_s(\mathcal{B}))^2 + (\beta_e(\mathcal{B}) - \alpha_e(\mathcal{B}))^2\right)^{1/2} \\ &\leq \left((\delta^*/\sqrt{2})^2 + (\delta^*/\sqrt{2})^2\right)^{1/2} \leq \delta^*. \end{aligned}$$

It then follows that for any $\mathcal{B} \subset \mathcal{D}$, $\text{size}(\mathcal{B}) \leq \delta$ implies

$$\beta(\mathcal{B}) - \alpha(\mathcal{B}) = l(\beta_s(\mathcal{B}), \alpha_e(\mathcal{B})) - l(\alpha_s(\mathcal{B}), \beta_e(\mathcal{B})) \leq \varepsilon$$

which proves (C2) for the probability of improvement cost function. The proof for the expected improvement cost function is similar: Let $\varepsilon > 0$ be arbitrary. As before,

we can view the expected improvement cost function as a function of $s(x)$ and $e(x)$ and we have from the monotonicity (6.1.7), (6.1.8) that it has bounded gradient since $\Phi(\cdot), \phi(\cdot) \leq 1$. It is therefore uniformly continuous and there exists $\delta^* > 0$ such that for any $\mathcal{B} \subset \mathcal{D}$

$$\|(\beta_s(\mathcal{B}), \alpha_e(\mathcal{B})) - (\alpha_s(\mathcal{B}), \beta_e(\mathcal{B}))\|_2 \leq \delta^* \implies l(\beta_s(\mathcal{B}), \alpha_e(\mathcal{B})) - l(\alpha_s(\mathcal{B}), \beta_e(\mathcal{B})) \leq \varepsilon$$

where $l(\cdot, \cdot)$ denotes the expected improvement cost function as a function of $s(x)$ and $e(x)$. The last part of the proof is then the same as for the probability of improvement cost function above and we obtain a suitable $\delta > 0$ such that for any $\mathcal{B} \subset \mathcal{D}$, $\text{size}(\mathcal{B}) \leq \delta$ implies

$$\beta(\mathcal{B}) - \alpha(\mathcal{B}) = l(\beta_s(\mathcal{B}), \alpha_e(\mathcal{B})) - l(\alpha_s(\mathcal{B}), \beta_e(\mathcal{B})) \leq \varepsilon$$

which proves (C2) for the expected improvement cost function. □

6.1.3 Numerical Example

Consider the problem of finding the global minimum of a cubic spline RBF surrogate $s(x, y)$ fitted to the Dixon-Szegő six hump camel back function (see Dixon and Szegő, 1978) at thirty scattered points in $[-2, 2] \times [-1.25, 1.25]$. The optimal solution as found in 1621 iterations of step 1 of the algorithm with a tolerance of 1×10^{-5} is shown in Figure 6.2 below. This took about 78 seconds of cpu time for a Matlab implementation on an AMD Phenom II X3 705e processor machine.

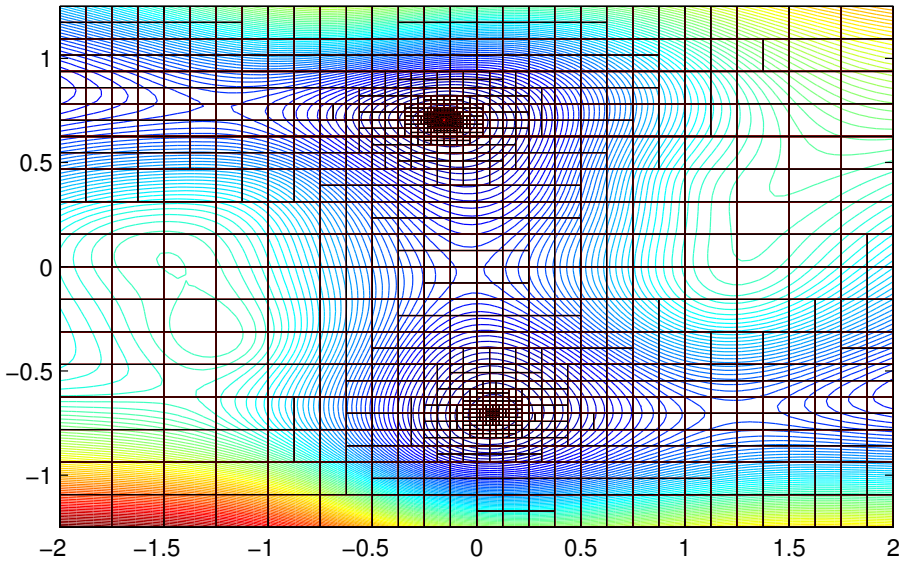

Figure 6.2: *Contours of the RBF surrogate $s(x, y)$ fitted to the camel function. The black rectangles denote the boxes $\mathcal{B}$ used by the branch and bound algorithm. Note that they cluster at the global minimum of $s(x, y)$ which is denoted by a red circle.*

6.2 Convex Constraints

In this section we present a branch and bound algorithm for globally optimizing the cost function $l(x)$ subject to convex constraints, based on the canonical branch and bound algorithm for bound constrained global optimization from Section 6.1.

6.2.1 Description of the Algorithm

We extend the canonical branch and bound algorithm from Section 6.1 (using ideas from Pedamallu et al., 2008) to the case where we have convex constraints, i.e.

$$\mathcal{D} = \{x \in \mathbb{R}^N : c(x) \leq 0\}$$

where $c : \mathbb{R}^N \to \mathbb{R}$ is a convex function and we assume that $\mathcal{D} \neq \emptyset$ is bounded. We calculate the componentwise extrema l_i and u_i of the domain $\mathcal{D}$ by solving the $2N$ convex optimization problems

$$\begin{aligned} &\min \pm e_i^T x \quad i = 1, \ldots, N \\ &\text{s.t. } c(x) \leq 0 \end{aligned}$$

where e_i denotes the i-th unit vector in $\mathbb{R}^N$. Then we can construct the minimal bounding box $\mathcal{B}_0 = [l_1, u_1] \times \ldots \times [l_N, u_N]$ for the domain $\mathcal{D}$. Note that these are inexpensive global optimization problems which are almost always solvable in polynomial time using interior point methods (see Boyd and Vandenberghe, 2004, Section 1.3.1). Before describing the algorithm in detail we first need some notation. Let $\mathcal{B} = [l^{\mathcal{B}}, u^{\mathcal{B}}]$ denote the N-dimensional interval $[l_1^{\mathcal{B}}, u_1^{\mathcal{B}}] \times \ldots \times [l_N^{\mathcal{B}}, u_N^{\mathcal{B}}]$. Define the condition number of $\mathcal{B}$ as the ratio of the largest to the smallest edge length, that is

$$\text{cond}(\mathcal{B}) = \frac{\max_i(u_i^{\mathcal{B}} - l_i^{\mathcal{B}})}{\min_i(u_i^{\mathcal{B}} - l_i^{\mathcal{B}})}$$

and the size of $\mathcal{B}$ as half the largest edge length, that is

$$\text{size}(\mathcal{B}) = \frac{\max_i(u_i^{\mathcal{B}} - l_i^{\mathcal{B}})}{2}.$$

As before, let $\alpha(\mathcal{B}), \beta(\mathcal{B})$ denote lower and upper bounds for the global minimum of the cost function $l(x)$ over a box $\mathcal{B} \subset \mathcal{B}_0$ satisfying the conditions (C1),(C2) (see Section 6.1). But this time let $x_{\mathcal{B}}$ from the branch and bound algorithm (p. 66) in (6.1.2) and (6.1.4) denote a feasible vertex of $\mathcal{B}$, i.e. a vertex of $\mathcal{B}$ which lies in $\mathcal{D}$, unless $\mathcal{B}$ has been locally searched, in which case it is the feasible point found by a constrained local search procedure (see step 1f in the algorithm below). Note that boxes with infeasible vertices are discarded by the algorithm (see step 1d) and all boxes in the initial list $\mathcal{L}_0$ must have feasible vertices as they are formed by bisecting the initial bounding box $\mathcal{B}_0$ along each coordinate (see step 0b).

Once again, the idea behind the algorithm is to recursively partition the bounding box $\mathcal{B}_0$ into sub-boxes until we find a box of sufficiently small size containing the global minimum

of $l(x)$ over $\mathcal{D}$. Since we are able to obtain bounds on the minimum of $l(x)$ over any box in $\mathcal{B}_0$, we can use them to discard boxes which cannot contain the global minimum. Moreover, we can also discard boxes which are infeasible with respect to the constraints, i.e. boxes which do not contain a feasible region. As before, we accelerate the algorithm by running constrained local searches on all previously unsearched boxes if no boxes are discarded after two successive iterations of the algorithm. The algorithm proceeds as follows, cf. the branch and bound algorithm in Section 6.1:

Branch and Bound Algorithm for Convex Constraints

0. Initialisation:

a) Set $k = 0$ and $s = 0$.

b) Bisect the initial bounding box $\mathcal{B}_0$ into 2^N sub-boxes $\mathcal{B}_1, \ldots, \mathcal{B}_{2^N}$ (once along each coordinate) and set $x_{\mathcal{B}_i}$ to be a feasible vertex for each of the sub-boxes $\mathcal{B}_i$.

c) Let $\mathcal{L}_0 = \{\mathcal{B}_1, \ldots, \mathcal{B}_{2^N}\}$ be the initial list of boxes.

d) Let $U_0 = \min_{\mathcal{B} \in \mathcal{L}_0} \beta(\mathcal{B})$ be the initial upper bound for $\min_{x \in \mathcal{D}} l(x)$.

e) Let $L_0 = \min_{\mathcal{B} \in \mathcal{L}_0} \alpha(\mathcal{B})$ be the initial lower bound for $\min_{x \in \mathcal{D}} l(x)$.

1. While $U_k - L_k > \varepsilon$, repeat the following procedure:

a) Remove from $\mathcal{L}_k$ boxes $\mathcal{B} \in \mathcal{L}_k$ such that $\alpha(\mathcal{B}) > U_k$.

b) Choose $\mathcal{B} \in \mathcal{L}_k$ such that $\alpha(\mathcal{B}) = L_k$.

c) If $\mathcal{B}$ was formed by splitting its parent box along the i-th coordinate, bisect $\mathcal{B}$ along the $i + 1$ mod N coordinate into $\mathcal{B}_I$ and $\mathcal{B}_{II}$. Set $\mathcal{L}_{k+1} := \mathcal{L}_k \setminus \{\mathcal{B}\}$.

d) Check if the vertices of $\mathcal{B}_I, \mathcal{B}_{II}$ satisfy the constraints. For $i = I$ and II proceed as follows: If $\mathcal{B}_i$ has a feasible vertex, set $x_{\mathcal{B}_i}$ to be that vertex and set $\mathcal{L}_{k+1} = \mathcal{L}_{k+1} \cup \{\mathcal{B}_i\}$. If on the other hand none of the vertices of $\mathcal{B}_i$ are feasible, discard $\mathcal{B}_i$.

e) If any boxes have been discarded in 1.a) set $s = 0$, otherwise set $s = s + 1$.

f) If $s > 2$ run an approximate constrained local search algorithm on all previously unsearched boxes $\mathcal{B}$ in $\mathcal{L}_{k+1}$, update $x_{\mathcal{B}}$ to be the relevant feasible point found by the algorithm and set $s = 0$.

g) Set $U_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \beta(\mathcal{B})$.

h) Set $L_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \alpha(\mathcal{B})$.

i) Set $k = k + 1$.

2. Return U_k as the estimate of the global minimum of $l(x)$ over $\mathcal{D}$.

6.2.2 Proof of Convergence

We will prove that, under suitable assumptions, the above algorithm converges in a finite number of iterations to within a tolerance $\varepsilon > 0$ of the global minimum of $l(x)$. First, we state and prove the following Lemma which follows from our splitting rule, i.e. the requirement that we cycle the coordinate along which we bisect a box in step $1c$ of the algorithm.

Lemma 6.9. *For any $k \in \mathbb{N}$ and for any box $\mathcal{B} \in \mathcal{L}_k$,*

$$cond(\mathcal{B}) \leq 2\, cond(\mathcal{B}_0).$$

Proof: Consider a box $\mathcal{B} = [l^{\mathcal{B}}, u^{\mathcal{B}}] \in \mathcal{L}_k$ and let $e_{max} = \max_i(u_i^{\mathcal{B}} - l_i^{\mathcal{B}}), e_{min} = \min_i(u_i^{\mathcal{B}} - l_i^{\mathcal{B}})$. When we bisect this box into two child boxes $\mathcal{B}_I$ and $\mathcal{B}_{II}$ we do so in one of the following ways:

1. Split along an edge that does not have length e_{max} or e_{min}. Then

$$\text{cond}(\mathcal{B}_i) = \frac{e_{max}}{e_{min}} = \text{cond}(\mathcal{B})$$

   for $i = I$ and II.

2. Split along an edge with length e_{min}. Then

$$\text{cond}(\mathcal{B}_i) = \frac{e_{max}}{e_{min}/2} = 2\,\text{cond}(\mathcal{B})$$

   for $i = I$ and II.

3. Split along an edge with length e_{max}. Then

   a) if $e_{min} \leq e_{max}/2$,

$$\text{cond}(\mathcal{B}_i) = \frac{e_{max}}{e_{min}} = \text{cond}(\mathcal{B})$$

      for $i = I$ and II.

   b) if $e_{min} > e_{max}/2$,

$$\text{cond}(\mathcal{B}_i) = \frac{e_{max}}{e_{max}/2} = 2 \leq 2\,\text{cond}(\mathcal{B})$$

      for $i = I$ and II, as $\text{cond}(\mathcal{B}) \geq 1$ (since $e_{max} \geq e_{min} \Rightarrow e_{max}/e_{min} \geq 1$).

Therefore, the upper bound on the condition number is given by splitting along an edge with largest or smallest length. As our splitting rule cycles through the coordinate to bisect along, all the edges of a box must be split before we can split along an edge with largest or smallest length again. So, if we are to split $\mathcal{C} = \mathcal{B}_I \in \mathcal{L}_{k+1}$ into two child boxes $\mathcal{C}_I$ and $\mathcal{C}_{II}$ along an edge with largest or smallest length again, we have

$$\text{cond}(\mathcal{C}) = \frac{e_{max}/2}{e_{min}/2}.$$

Thus, if we split along an edge with smallest length ($e_{min}/2$), we get that

$$\text{cond}(\mathcal{C}_i) = \frac{e_{max}/2}{e_{min}/4} = 2\,\frac{e_{max}}{e_{min}} = 2\,\text{cond}(\mathcal{B})$$

for $i = I$ and II. Similarly, if we split along an edge with largest length ($e_{max}/2$) assuming $e_{min} > e_{max}/2$, we get that

$$\text{cond}(\mathcal{C}_i) = \frac{e_{max}/2}{e_{max}/4} = 2 \leq 2\,\text{cond}(\mathcal{B})$$

for $i = I$ and II. By induction we obtain that we can bound the condition number of any box $\mathcal{B} \in \mathcal{L}_k$ for any k by $2\,\text{cond}(\mathcal{B}_0)$. □

We now proceed with the main part of the proof, that the proposed algorithm converges in a finite number of iterations to within a tolerance $\varepsilon > 0$ of the global minimum of $l(x)$ over $\mathcal{D}$. Formally:

Theorem 6.10. *For any $\varepsilon > 0$ there exists $M_\varepsilon \in \mathbb{N}$ such that*

$$U_{M_\varepsilon} - L_{M_\varepsilon} \leq \varepsilon$$

and U_{M_ε} is within a tolerance ε of the global minimum of $l(x)$ over $\mathcal{D}$.

Proof: We follow the proof given in the appendix of Balakrishnan et al. (1991). Whenever a parent box is split into two child boxes in step 1c of the algorithm, one of the child boxes must have at least one feasible vertex and so cannot be discarded. (This is because the parent box has at least one feasible vertex as otherwise it would have been discarded in step 1d.) Since our initial partition $\mathcal{L}_0$ has 2^N boxes of equal volume $\text{vol}(\mathcal{B}_0)/2^N$, it is easy to see that after k iterations

$$\min_{\mathcal{B}\in\mathcal{L}_k} \text{vol}(\mathcal{B}) \leq \frac{\text{vol}(\mathcal{B}_0)}{2^N + k} \tag{6.2.1}$$

as the partition $\mathcal{L}_k$ has at most $2^N + k$ boxes. Now observe that

$$\begin{aligned}
\text{vol}(\mathcal{B}) = \prod_{i=1}^{N}(u_i - l_i) &\geq \max_i(u_i - l_i)\left(\min_i(u_i - l_i)\right)^{N-1} \\
&= \frac{(2\ \text{size}(\mathcal{B}))^N}{\text{cond}(\mathcal{B})^{N-1}} \\
&\geq \left(\frac{2\ \text{size}(\mathcal{B})}{\text{cond}(\mathcal{B})}\right)^N \\
\implies \quad \text{size}(\mathcal{B}) &\leq 1/2\ \text{cond}(\mathcal{B})\text{vol}(\mathcal{B})^{1/N}.
\end{aligned} \tag{6.2.2}$$

Combining Lemma 6.9 with inequalities (6.2.1) and (6.2.2) gives

$$\min_{\mathcal{B}\in\mathcal{L}_k} \text{size}(\mathcal{B}) \leq \text{cond}(\mathcal{B}_0)\left(\frac{\text{vol}(\mathcal{B}_0)}{2^N + k}\right)^{1/N}. \tag{6.2.3}$$

Let $\mathcal{B}_k := \arg\min_{\mathcal{B}\in\mathcal{L}_k} \text{size}(\mathcal{B})$ and let $\mathcal{A}_k \in \mathcal{L}_{M_k}$ be the box which for some $M_k < k$ we split to obtain $\mathcal{B}_k$. Let $\varepsilon > 0$ be arbitrary. Then there exists $\delta > 0$ such that for any $\mathcal{B} \subset \mathcal{B}_0$,

$$\text{size}(\mathcal{B}) \leq 2\delta \implies \beta(\mathcal{B}) - \alpha(\mathcal{B}) \leq \varepsilon \tag{6.2.4}$$

by assumption (C2). Choose $K \in \mathbb{N}$ sufficiently large such that

$$\text{size}(\mathcal{B}_K) \leq \delta$$

which is possible by (6.2.3). Then $\mathcal{A}_K$ must have $\text{size}(\mathcal{A}_K) \leq 2\delta$ and thus from (6.2.4) we have

$$\beta(\mathcal{A}_K) - \alpha(\mathcal{A}_K) \leq \varepsilon. \tag{6.2.5}$$

Now, as $\mathcal{A}_K$ was split at iteration M_K, it must have satisfied $\alpha(\mathcal{A}_K) = L_{M_K}$. Hence, we get that

$$U_{M_K} - L_{M_K} \leq \beta(\mathcal{A}_K) - L_{M_K} \leq \varepsilon \tag{6.2.6}$$

since $U_{M_K} \leq \beta(\mathcal{A}_K)$ by definition and using (6.2.5). We therefore have an upper bound M_K on the number of branch and bound iterations.

It remains to show that U_{M_K} is within a tolerance ε of the global minimum of $l(x)$ over the convex set $\mathcal{D}$. Assume condition (C1) holds and suppose that the global minimum l^* of $l(x)$ over $\mathcal{D}$ is attained at $x^* \in \mathcal{D}$. First, we show that x^* is contained in a box in $\mathcal{L}_{M_K}$. To see this, observe that for all $k \in \mathbb{N}$, $\mathcal{L}_k$ is a partition of the bounding box $\mathcal{B}_0$ with boxes which cannot possibly contain x^* removed, that is to say boxes $\mathcal{B}$ which have lower bound

$$\alpha(\mathcal{B}) > U_k$$

i.e. $\alpha(\mathcal{B}) > l(x_\mathcal{B})$ for a feasible vertex $x_\mathcal{B} \in \mathcal{D}$. As x^* is contained in a box in $\mathcal{L}_{M_K}$ it follows that $L_{M_K} \leq l^*$ and thus

$$U_{M_K} - l^* \leq U_{M_K} - L_{M_K} \leq \varepsilon$$

by (6.2.6). □

6.2.3 Numerical Example

Consider again the problem of finding the minimum of the RBF surrogate $s(x, y)$ fitted to the Dixon-Szegő six hump camel back function, this time subject to the convex constraint

$$c(x, y) = \begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - 1 \leq 0$$

which is an ellipse. The optimal solution as found in 1418 loops of step 1 of the algorithm with a tolerance of 1×10^{-5} is shown in Figure 6.3 below. This took about 62 seconds of cpu time.

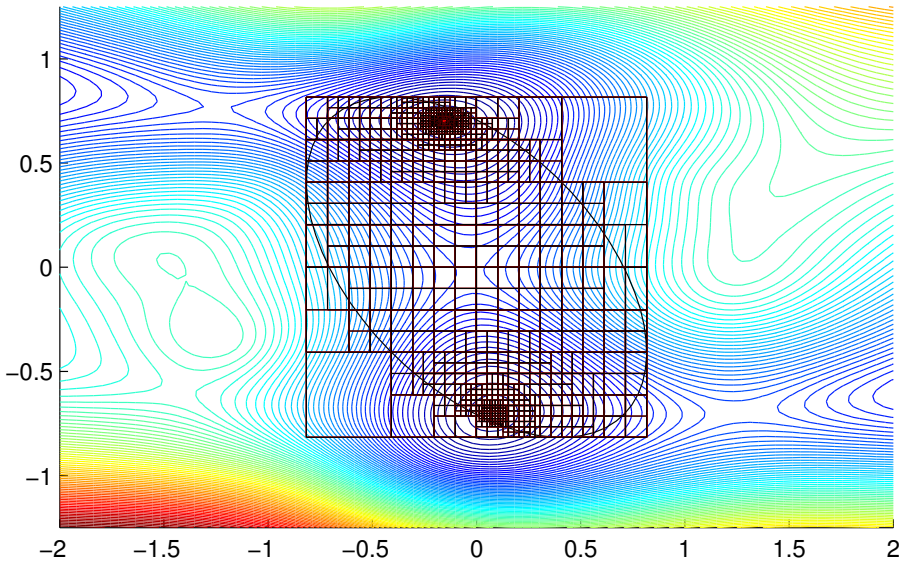

Figure 6.3: *Contours of the RBF surrogate $s(x, y)$ fitted to the camel function. The black squares denote the boxes $\mathcal{B}$ used by the branch and bound algorithm. Note that they cluster at the global minimum of $s(x, y)$ which is denoted by a red circle.*

6.3 Mixed Integer Constraints

In this section we will look at extending the branch and bound algorithm for convex constraints from Section 6.2 to integer constraints. Formally, we consider the mixed integer global optimization problem

$$\begin{aligned} &\min_{(x,y)\in\mathcal{D}\times\mathcal{I}} l(x, y) \\ &\text{s. t. } y \in \mathcal{I} \cap \mathbb{Z}^m \end{aligned}$$

where $l : \mathcal{D} \times \mathcal{I} \subset \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a continuous cost function. As in Section 6.2 we have convex constraints

$$\mathcal{D} = \{x \in \mathbb{R}^n : c(x) \leq 0\}$$

(where $c : \mathbb{R}^n \to \mathbb{R}$ is a convex function) as well as a small ($m < 20$) number of integer constraints $y \in \mathcal{I} \cap \mathbb{Z}^m$ where

$$\mathcal{I} = \{y \in \mathbb{R}^m : a \leq y \leq b\}$$

for some $a, b \in \mathbb{Z}^m$ with $a < b$. The approach we take is to combine the standard branch and bound algorithm for integer programming with our branch and bound algorithm for convex constraints from Section 6.2. Following ideas from the generalised Benders' decomposition (see p. 1118 in Floudas, Aggarwal and Ciric, 1989), we separate the optimization problem into an equivalent form consisting of an outer and inner optimization problem:

$$\begin{aligned} &\min_{y\in\mathcal{I}} v(y) \\ &\text{s. t. } y \in \mathcal{I} \cap \mathbb{Z}^m \end{aligned} \tag{6.3.1}$$

where

$$v(y) = \left\{ \min_{x \in \mathcal{D}} l(x, y) : c(x) \leq 0 \right\}. \tag{6.3.2}$$

We can then solve the inner optimization problem (6.3.2) for fixed y and real intervals over y using our algorithm from Section 6.2. The outer optimization problem (6.3.1) can be solved using the standard integer programming branch and bound algorithm (see Section 7.4 in Wolsey, 1998) where instead of solving linear programming or Lagrangian relaxations, we solve (6.3.2) for relaxed values $y \in \mathcal{I}$. Thus we solve the inner problem (6.3.2) for all continuous $y \in \mathcal{I}$ and branch on integers y to selectively fix the y variables. The inner problem (6.3.2) is then solved again for all continuous y in $\mathcal{I}$ except the ones which have been fixed as integers and the procedure is repeated. Lower bounds are given by solutions to (6.3.2) for relaxed y and upper bounds are given by these solutions if the minimisers y are integers, or by convention infinity if they are not integers (see step 1c in the algorithm below). The complete algorithm proceeds as follows (cf. the canonical branch and bound algorithm in Section 6.1 and the integer programming branch and bound algorithm in Section 7.4 of Wolsey, 1998):

Branch and Bound Algorithm for Mixed Integer Constraints

0. Initialisation:

a) Set $k = 0$.

b) Let $\mathcal{B}_0 = \mathcal{D} \times \mathcal{I}$ *be the initial box and* $\mathcal{L}_0 = \{\mathcal{B}_0\}$ *the initial list of boxes.*

c) Solve the relaxed problem $\min_{(x,y)} l(x, y)$ *over* $\mathcal{B}_0$ *using our algorithm from Section 6.2 to obtain a minimiser* $(x_{\mathcal{B}_0}, y_{\mathcal{B}_0})$. *If* $y_{\mathcal{B}_0}$ *is an integer, let* $\beta(\mathcal{B}_0) = l(x_{\mathcal{B}_0}, y_{\mathcal{B}_0})$, *otherwise let* $\beta(\mathcal{B}_0) = \infty$. *Let* $\alpha(\mathcal{B}_0) = l(x_{\mathcal{B}_0}, y_{\mathcal{B}_0})$.

d) Let $L_0 = \alpha(\mathcal{B}_0)$ *be the initial lower bound for* $\min_{(x,y) \in \mathcal{D} \times \mathcal{I} \cap \mathbb{Z}^m} l(x, y)$.

e) Let $U_0 = \beta(\mathcal{B}_0)$ *be the initial upper bound for* $\min_{(x,y) \in \mathcal{D} \times \mathcal{I} \cap \mathbb{Z}^m} l(x, y)$.

1. While $U_k - L_k > 0$, *repeat the following procedure:*

a) Remove from $\mathcal{L}_k$ *boxes* $\mathcal{B} \in \mathcal{L}_k$ *such that* $\alpha(\mathcal{B}) > U_k$.

b) Choose $\mathcal{B} \in \mathcal{L}_k$ *such that* $\alpha(\mathcal{B}) = L_k$.

c) Choose an integer variable y^j *for which* $y^j_{\mathcal{B}}$ *is not an integer and bisect* $\mathcal{B}$ *into* $\mathcal{B}_I = \mathcal{B} \cap \mathcal{D} \times \{y : y^j \leq \lfloor y^j_{\mathcal{B}} \rfloor\}$ *and* $\mathcal{B}_{II} = \mathcal{B} \cap \mathcal{D} \times \{y : y^j \geq \lceil y^j_{\mathcal{B}} \rceil\}$. *Set* $\mathcal{L}_{k+1} := \mathcal{L}_k \cup \{\mathcal{B}_I, \mathcal{B}_{II}\}$ *and remove* $\mathcal{B}$ *from* $\mathcal{L}_{k+1}$.

d) For $i = I$ *and* II *proceed as follows: Solve the relaxed problem* $\min_{(x,y)} l(x, y)$ *over* $\mathcal{B}_i$ *using our algorithm from Section 6.2 to obtain a minimiser* $(x_{\mathcal{B}_i}, y_{\mathcal{B}_i})$. *If* $y_{\mathcal{B}_i}$ *is an integer, let* $\beta(\mathcal{B}_i) = l(x_{\mathcal{B}_i}, y_{\mathcal{B}_i})$, *otherwise let* $\beta(\mathcal{B}_i) = \infty$. *Let* $\alpha(\mathcal{B}_i) = l(x_{\mathcal{B}_i}, y_{\mathcal{B}_i})$.

e) Set $L_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \alpha(\mathcal{B})$.

f) Set $U_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \beta(\mathcal{B})$.

g) Set $k = k + 1$.

2. *Return* U_k *as the estimate of the global minimum of* $l(x, y)$ *over* $\mathcal{D} \times \mathcal{I} \cap \mathbb{Z}^m$.

Note that the algorithm converges to the global minimum of $l(x, y)$ as both the integer programming and convex branch and bound algorithms are globally convergent.

6.3.1 Numerical Example

As before, consider the problem of finding the global minimum of a cubic spline RBF surrogate $s(x, y)$ fitted to the Dixon-Szegő six hump camel back function at thirty scattered points in $[-2, 2] \times [-1.25, 1.25]$. Except this time we constrain the x variable to only take integer values, i.e. $x \in [-2, 2] \cap \mathbb{Z}$. The optimal solution is shown in Figure 6.4 below. For

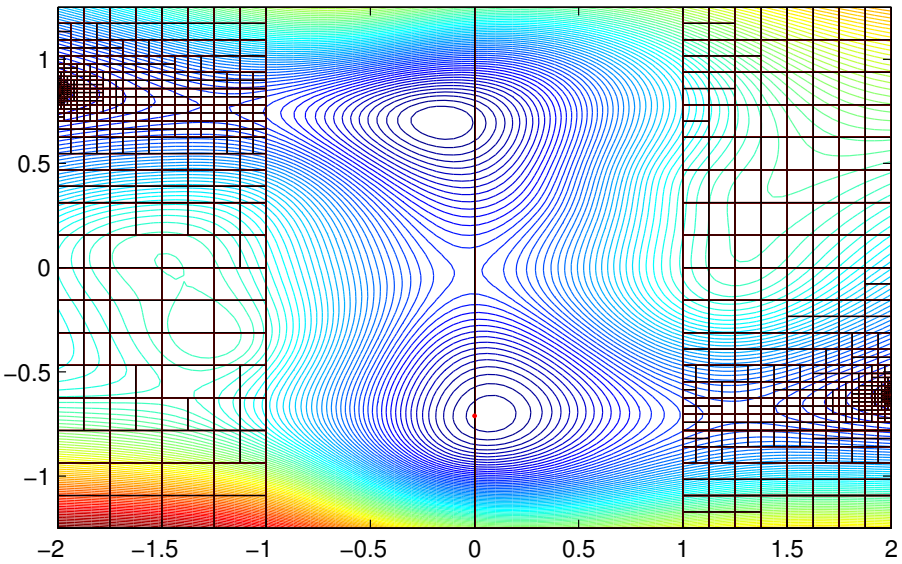

Figure 6.4: *Contours of the RBF surrogate* $s(x, y)$ *fitted to the camel function. The black rectangles denote the boxes used by the branch and bound algorithm when performing the inner optimization* (6.3.2) *on the regions* $[-2, -1], 0, [1, 2] \times [-1.25, 1.25]$. *Previous inner optimizations on the regions* $[-2, 2], [0, 2] \times [-1.25, 1.25]$ *have been omitted from the figure for clarity. The constrained global minimum of* $s(x, y)$ *is denoted by a red circle.*

a more interesting example, consider the problem of finding the global minimum of a cubic spline RBF surrogate $s(x, y, z)$ fitted to the Hartman 3 function (Dixon and Szegő, 1978) scaled to $[0, 10]^3$ at thirty scattered points. This time we constrain the y variable to only take integer values. A 3D slice view of the surrogate $s(x, y, z)$ is shown in Figure 6.5. The optimal value of -3.690155 at $(0, 6, 8.132228)$ was found after performing only three inner optimizations on the regions $[0, 10]^3, [0, 10] \times [0, 5] \times [0, 10]$ and $[0, 10] \times [6, 10] \times [0, 10]$.

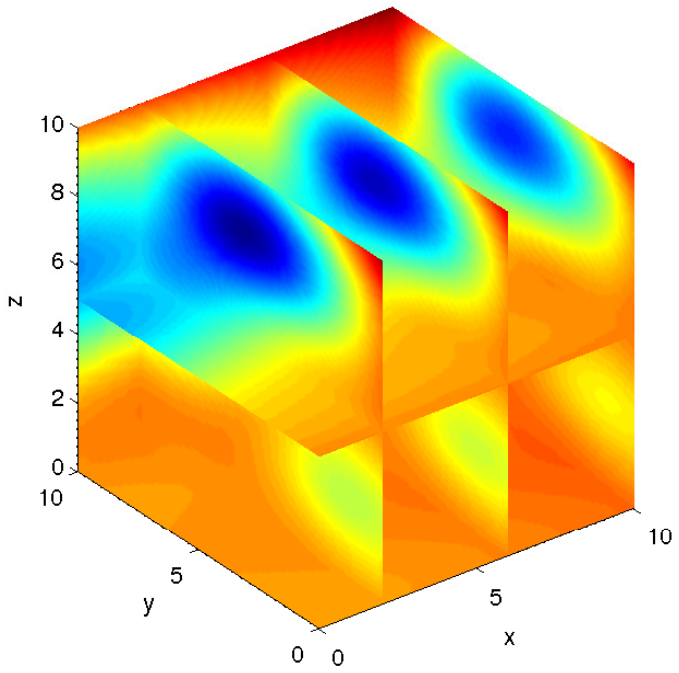

Figure 6.5: *3D slices of the RBF surrogate $s(x, y, z)$ fitted to the Hartman 3 function.*

6.4 Triangular Meshes and Spherical Surfaces

So far we have considered finding the global minimum of the surrogate over rectangular or convex regions. In this section we will extend this to more general, potentially disconnected, regions. Many applications of Numerical Analysis involving the solution of partial differential equations make use of two and three dimensional triangular meshes of the domains of interest. With this in mind, we will look at a branch and bound algorithm for optimising our cost functions on such triangular meshes and through the use of meshes extend the algorithm to the surface of a sphere.

6.4.1 Triangular Meshes

In this subsection we assume that the domain $\mathcal{D}$ of our expensive objective function f is a triangular mesh in $\mathbb{R}^2$ or $\mathbb{R}^3$. The objective function may be defined outside of the triangular mesh, in which case we would have the constraint set $\mathcal{C}$ consisting of the mesh in place of $\mathcal{D}$. Either way, we can fit a two or three dimensional surrogate to the objective function which enables us to construct any of the cost functions $l(x)$ as detailed in Section 5.2. First, define a triangle $\mathcal{T} \subset \mathbb{R}^n$ as the convex hull of its vertices $v_1, \ldots, v_{n+1} \in \mathbb{R}^n$

$$\mathcal{T} = \left\{ \sum_{k=1}^{n+1} \xi_k v_k : \sum_{k=1}^{n+1} \xi_k = 1,\ \xi_k \geq 0\ \forall k \right\}$$

where $n = 2$ or 3. Also, let size($\mathcal{T}$) denote the maximum half-length of the sides of $\mathcal{T}$. Since the domain $\mathcal{D}$ is now partitioned into triangles as opposed to boxes we need to calculate the bounds α, β for the various cost functions from Section 6.1 over a triangle $\mathcal{T} \subset \mathcal{D}$. As in Section 6.1, it suffices to specify bounds on the surrogate $s(x)$ and the error $e(x)$. As before, we bound each radial basis function term in the surrogate over a triangle $\mathcal{T} \subset \mathcal{D}$ using quadratic functions

$$a_j + b_j\|x - x_j\|_W^2 \leq \varphi(\|x - x_j\|_W) \leq A_j + B_j\|x - x_j\|_W^2.$$

Once again, it is easy to obtain exact bounds l_j, u_j on $r_j = \|x - x_j\|_W^2$ over a triangle $\mathcal{T}$. This is because u_j is always attained at one of the vertices of the triangle $\mathcal{T}$ and l_j is attained at the closest point in $\mathcal{T}$ to x_j. Finding the closest point to a triangle in $2D$ can be done analytically and in $3D$ amounts to solving the convex optimization problems

$$\begin{aligned} \min_{\xi \in \mathbb{R}^4} & \left\| \sum_{k=1}^{4} \xi_k v_k - x_j \right\|_W^2 \\ \text{s.t. } & \sum_{k=1}^{4} \xi_k = 1, \\ & \xi_k \geq 0 \ \forall k. \end{aligned}$$

The coefficients a_j, b_j, A_j, B_j are then defined as in Section 6.1. We can now define the lower bounding function for the global minimum of the surrogate over $\mathcal{T}$ as

$$\alpha_s(\mathcal{T}) = \min_{x \in \mathcal{T}} \{p_s(x) + c_s(x)\} \tag{6.4.1}$$

where

$$p_s(x) = \sum_{k=1}^{M} \mu_k \pi_k(x)$$

as before and

$$c_s(x) := q_s(x) + \gamma(x - \underline{x})^T(x - \overline{x})$$

is the convex relaxation of

$$q_s(x) = \sum_{\substack{j=1 \\ \lambda_j > 0}}^{N} \lambda_j(a_j + b_j\|x - x_j\|_W^2) + \sum_{\substack{j=1 \\ \lambda_j < 0}}^{N} \lambda_j(A_j + B_j\|x - x_j\|_W^2),$$

where $\underline{x} = \min_{x \in \mathcal{T}} x$ and $\overline{x} = \max_{x \in \mathcal{T}} x$, which can easily be minimised componentwise. The relaxation parameter $\gamma = \max\{0, -\lambda_{min}\}$ where λ_{min} is the smallest eigenvalue of the Hessian matrix of $q_s(x)$. Note that the Hessian matrix of $q(x)$ is $2(\sum_{\lambda_j > 0} \lambda_j b_j + \sum_{\lambda_j < 0} \lambda_j B_j)W^T W$ where the weight matrix W is diagonal and therefore the smallest eigenvalue is trivial to find. Further, using the cubic spline RBF means that $p_s(x)$ is a linear polynomial for which it is easy to obtain the minimum over a triangle. Finding the minimum

of $c_s(x)$ over $\mathcal{T}$ is a convex QP which we can easily solve. Similarly to before, the upper bounding function for the global minimum of the surrogate over $\mathcal{T}$ is given by

$$\beta_s(\mathcal{T}) = s(x_{\mathcal{T}}) \tag{6.4.2}$$

where $x_{\mathcal{T}}$ is the centroid of the triangle $\mathcal{T}$, i.e. if $\mathcal{T}$ has vertices v_1, v_2, v_3 then $x_{\mathcal{T}} = 1/2(v_1 + v_2 + v_3)$. For the error $e(x)$, the convex relaxation suggested in Section 6.1 also applies to the case of triangles. We are now in a position to apply the branch and bound algorithm in Section 6.1 to triangles $\mathcal{T}$ in the triangular mesh $\mathcal{D}$. The algorithm is essentially the same except that we start off with a triangulation of $\mathcal{D}$ as opposed to a single triangle. Also, each triangle is split into two triangles along the line going from the midpoint of the longest edge to the opposite vertex. The convergence proof for the algorithm remains virtually unchanged, we first give the following two Lemmas which are analogues of Lemmas 6.3 and 6.4 for triangles:

Lemma 6.11. *We have that*

$$\forall \varepsilon > 0 \; \exists \delta > 0 \; \textit{s.t.} \; \forall \mathcal{T} \subset \mathcal{D} \; \forall x, y \in \mathcal{T}, \; \textit{size}(\mathcal{T}) \leq \delta \implies r_s(x) - q_s(y) \leq \varepsilon$$

Proof: Identical to the proof of Lemma 6.3. □

Lemma 6.12. *Let $p \in \Pi_d^n$ be a polynomial on a compact set $\mathcal{S} \subset \mathbb{R}^n$. We then have that*

$$\forall \varepsilon > 0 \; \exists \delta > 0 \; \textit{s.t.} \; \forall \mathcal{T} \subset \mathcal{S} \; \forall x, y \in \mathcal{T}, \; \textit{size}(\mathcal{T}) \leq \delta \implies |p(x) - p(y)| \leq \varepsilon$$

Proof: Identical to the proof of Lemma 6.4. □

These lead to the following Theorem, which is essentially Theorem 6.5 for triangles:

Theorem 6.13. *The bounds α_s and β_s given above in* (6.4.1) *and* (6.4.2) *respectively satisfy the following two conditions:*

(C1s) $\alpha_s(\mathcal{T}) \leq \min_{x \in \mathcal{T}} s(x) \leq \beta_s(\mathcal{T})$

(C2s) $\forall \varepsilon > 0 \; \exists \delta > 0 \; \textit{s.t.} \; \forall \mathcal{T} \subset \mathcal{D}, \; \textit{size}(\mathcal{T}) \leq \delta \implies \beta_s(\mathcal{T}) - \alpha_s(\mathcal{T}) \leq \varepsilon$

Proof: (C1s) Clearly, the value of $s(\cdot)$ at any point in $\mathcal{T}$ is an upper bound on the minimum of $s(\cdot)$ over $\mathcal{T}$. Also, by construction $p_s(x) + c_s(x)$ underestimates $s(x)$ over $\mathcal{T}$ and thus $\alpha_s(\mathcal{T}) = \min_{x \in \mathcal{T}} \{p_s(x) + c_s(x)\}$ underestimates $\min_{x \in \mathcal{T}} s(x)$.

(C2s) Let $\varepsilon > 0$ be arbitrary. For clarity of exposition define for all $\mathcal{T} \subset \mathcal{D}$,

$$x_{\mathcal{T}}^* := \arg\min_{x \in \mathcal{T}} \{p_s(x) + q_s(x)\}$$

$$y_{\mathcal{T}}^* := \arg\min_{x \in \mathcal{T}} \{p_s(x) + c_s(x)\}$$

and let c denote some point in $\mathcal{T}$ (not necessarily the centroid). We have from Lemma 6.11 that there exists $\delta_1 > 0$ such that for any $\mathcal{T} \subset \mathcal{D}$

$$\text{size}(\mathcal{T}) \leq \delta_1 \implies r_s(c) - q_s(x^*_{\mathcal{T}}) \leq \varepsilon/4.$$

As $q_s(x)$ is a polynomial, we have from Lemma 6.12 that there exists $\delta_2 > 0$ such that for any $\mathcal{T} \subset \mathcal{D}$

$$\text{size}(\mathcal{T}) \leq \delta_2 \implies q_s(x^*_{\mathcal{T}}) - q_s(y^*_{\mathcal{T}}) \leq \varepsilon/4.$$

Note that as $\text{size}(\mathcal{T}) \to 0$ then by definition $\underline{x} \to x$ and $\overline{x} \to x$. Hence we have that there exists $\delta_3 > 0$ such that for any $\mathcal{T} \subset \mathcal{D}$

$$\text{size}(\mathcal{T}) \leq \delta_3 \implies -(y^*_{\mathcal{T}} - \underline{y^*_{\mathcal{T}}})^T(y^*_{\mathcal{T}} - \overline{y^*_{\mathcal{T}}}) \leq \varepsilon/4\gamma.$$

We also have from Lemma 6.12 that there exists $\delta_4 > 0$ such that for any $\mathcal{T} \subset \mathcal{D}$

$$\text{size}(\mathcal{T}) \leq \delta_4 \implies p_s(c) - p_s(y^*_{\mathcal{T}}) \leq \varepsilon/4.$$

So take $\delta := \min\{\delta_1, \delta_2, \delta_3, \delta_4\}$. We then have that for any $\mathcal{T} \subset \mathcal{D}$

$$\begin{aligned}
\text{size}(\mathcal{T}) \leq \delta \implies \beta_s(\mathcal{T}) - \alpha_s(\mathcal{T}) &= p_s(c) - p_s(y^*_{\mathcal{T}}) + r_s(c) - c_s(y^*_{\mathcal{T}}) \\
&= p_s(c) - p_s(y^*_{\mathcal{T}}) + r_s(c) - q_s(x^*_{\mathcal{T}}) + q_s(x^*_{\mathcal{T}}) - c_s(y^*_{\mathcal{T}}) \\
&= p_s(c) - p_s(y^*_{\mathcal{T}}) + r_s(c) - q_s(x^*_{\mathcal{T}}) + q_s(x^*_{\mathcal{T}}) - q_s(y^*_{\mathcal{T}}) \\
&\quad - \gamma(y^*_{\mathcal{T}} - \underline{y^*_{\mathcal{T}}})^T(y^*_{\mathcal{T}} - \overline{y^*_{\mathcal{T}}}) \\
&\leq \varepsilon/4 + \varepsilon/4 + \varepsilon/4 + \varepsilon/4 = \varepsilon.
\end{aligned}$$

□

We are now in a position to give the main theorem on the convergence of the branch and bound algorithm from Section 6.1 applied to triangles:

Theorem 6.14. *The bounds α and β given in* (6.1.9) − (6.1.13) *in Section 6.1 satisfy conditions (C1) and (C2) from that section for triangles. Hence, by the proof in Balakrishnan et al. (1991) the proposed algorithm converges in a finite number of iterations with an absolute accuracy $\varepsilon > 0$.*

Proof. Follows trivially from the analogue of Theorem 6.8 for triangles, using Theorem 6.13, the analogue of Theorem 6.7 for triangles and the definitions of α and β. □

6.4.2 Numerical Example

Consider the problem of finding the global minimum of a 20-point cubic spline RBF interpolant to the function

$$f(x) = \|x - (0, 0.125, 0)^T\|_W$$

with norm weights $w_i = 10$ over the 3-dimensional triangular mesh given by the Stanford Bunny. The Stanford Bunny is a canonical computer graphics test model available in a

number of resolutions (see Turk and Levoy, 1994). We will use the model with 948 triangles. The algorithm finds the global minimum in 79 iterations of step 1 the algorithm with a tolerance of 1×10^{-5}. This took about 12 seconds of cpu time. Figure 6.6 shows the mesh before and after the branch and bound algorithm has been run. Notice how some triangles have been split by the algorithm, particularly near the region containing the global minimum.

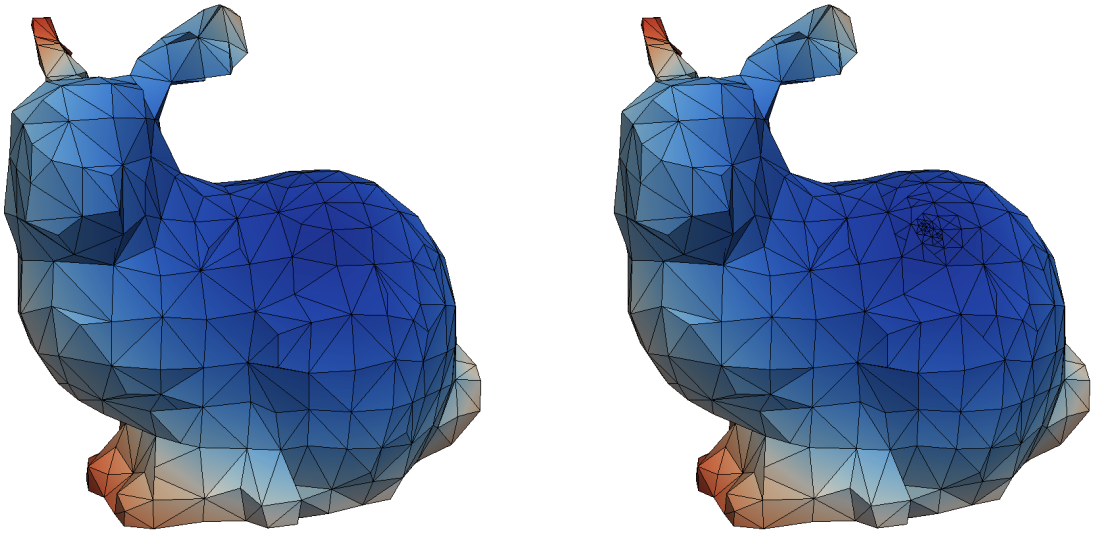

Figure 6.6: *Triangular mesh for the Stanford Bunny before (left) and after (right) running the branch and bound algorithm. The colours denote values of the RBF interpolant on the mesh.*

6.4.3 Surface of a Sphere

In this section we let the domain $\mathcal{D}$ of our expensive objective function be the surface of the n-dimensional unit sphere S^{n-1}. As before, we can fit an n-dimensional surrogate to our expensive function which enables us to construct any of the cost functions from Section 5.2. We optimize the cost function using a variant of the branch and bound algorithm from Section 6.1 which uses projected simplices as opposed to boxes. Let $\mathcal{T}$ be an n–1-simplex in $\mathbb{R}^n$ whose n vertices all lie on the unit sphere S^{n-1} and define the projected simplex $\mathcal{P}$ as the projection of $\mathcal{T}$ onto the unit sphere S^{n-1}

$$\mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b \text{ and } \|x\|_2 = 1\}$$

where the matrix inequality $Ax \leq b$ defines the half-spaces containing $\mathcal{P}$. These are given by calculating the convex hull of the origin 0 together with the n vertices of the simplex $\mathcal{T}$ and removing the half-space whose defining hyperplane does not go through the origin (see Figure 6.7 overleaf). We start the algorithm by generating a triangulation of random points on the sphere and projecting the simplices in this initial triangulation onto the sphere. This gives an initial partition of the sphere using projected simplices. We can then apply

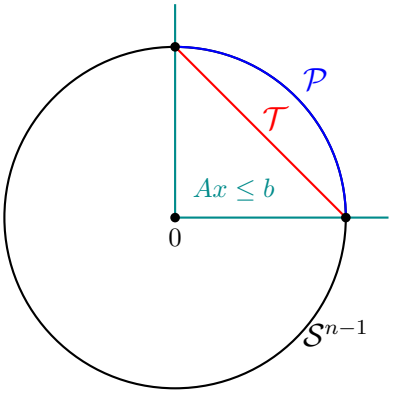

Figure 6.7: *A projected simplex $\mathcal{P}$ which is the projection of the simplex $\mathcal{T}$ onto $\mathcal{S}^{n-1}$ and is defined as the intersection of $\mathcal{S}^{n-1}$ with the region defined by $Ax \leq b$.*

the branch and bound algorithm from Section 6.1 to the projected simplices which remains virtually unchanged except for the splitting rule. Each simplex is split into two simplices along its longest edge. This is done by adding the midpoint of the longest edge as a vertex in addition to the existing vertices either side of the midpoint, thereby creating two new simplices which bisect the old one. Additionally we need upper and lower bounds on the surrogate $s(x)$ and associated error $e(x)$ over projected simplices $\mathcal{P}$. We proceed as in the previous subsection on triangular meshes, the only difference being how we calculate the exact bounds l_j, u_j on $r_j = \|x - x_j\|_W^2$ over a projected simplex $\mathcal{P}$ and the minimum of the convex relaxation

$$c(x) := q(x) + \gamma(x - \underline{x})^T(x - \overline{x})$$

over $\mathcal{P}$. This time $\underline{x} = \min_{x\in\mathcal{P}} x$ and $\overline{x} = \max_{x\in\mathcal{P}} x$ require solving the $2n$ componentwise optimization problems on the projected simplex

$$\begin{aligned} &\max_{x\in\mathbb{R}^n} / \min_{x\in\mathbb{R}^n} x_k \\ \text{s.t. } &Ax \leq b. \\ &\|x\|_2^2 = 1. \end{aligned}$$

for which we use an SQP method. Note that the optimization problems always have a unique global minimiser. To see this observe that the minimum of any of the above problems is also attained when the equality constraint $\|x\|_2^2 = 1$ is relaxed to one of $\|x\|_2^2 \leq 1$ or $\|x\|_2^2 \geq 1$ which gives a linear problem with a convex or concave constraint. Thus either the problem is convex or the optimum is attained at one of the vertices of $\mathcal{P}$. As before, the upper bound u_j is always attained at one of the vertices of the simplex $\mathcal{T}$ (whose projection is $\mathcal{P}$) and the lower bound l_j is attained at the closest point in $\mathcal{P}$ to $x_j \in S^{n-1}$. Finding the closest point amounts to solving the optimization problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \|x - x_j\|_W^2 \\ \text{s.t. } &Ax \leq b. \end{aligned}$$

$$\|x\|_2^2 = 1.$$

which we do using an SQP method. In this case, if the global minimum is not attained at one of the vertices of $\mathcal{P}$ it is unique. Finding the minimum of the convex relaxation $c(x)$ requires solving a convex optimization problem, the only difference being that $c(x)$ is now the objective function. The convergence proof for the algorithm remains virtually unchanged and we have the following theorem:

Theorem 6.15. *The bounds α and β given in (6.1.9) – (6.1.13) in Section 6.1 satisfy conditions (C1) and (C2) from that section for projected n–1-simplices. Hence, by the proof in Balakrishnan et al. (1991) the proposed algorithm converges in a finite number of iterations with an absolute accuracy $\varepsilon > 0$.*

Proof. Follows trivially from the analogue of Theorem 6.8 for projected simplices, using the analogues of Theorems 6.13, 6.7 for projected simplices and the definitions of α and β. □

6.4.4 Numerical Example

Consider the problem of finding the global minimum of a 20-point cubic spline RBF interpolant to the function

$$f(x) = -\exp(-(5\arccos(x_3))^2)$$

over the unit sphere in $\mathbb{R}^3$. The algorithm finds the global minimum in 130 iterations of step 1 of the algorithm with a tolerance of 1×10^{-5}. This took about 27 seconds of cpu time. While the algorithm makes use of projected triangles $\mathcal{P}$, these are difficult to visualise and so Figure 6.8 shows the mesh of underlying unprojected triangles $\mathcal{T}$, before and after the branch and bound algorithm has been run. Notice how a large number of triangles have been split by the algorithm, particularly near the region containing the global minimum.

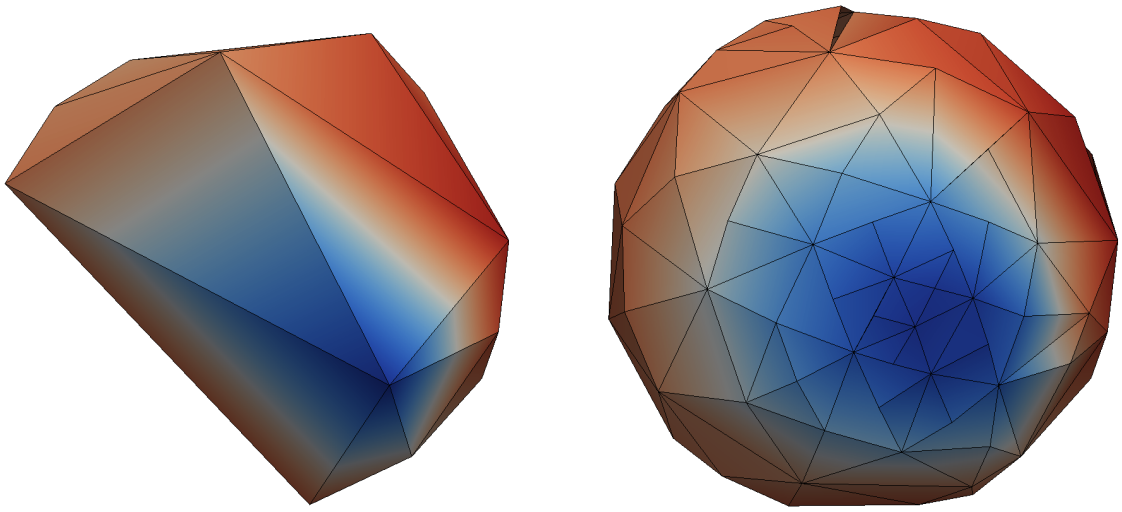

Figure 6.8: *Triangular mesh for the Sphere before (left) and after (right) running the branch and bound algorithm. The colours denote values of the RBF interpolant on the mesh.*

6.5 Lipschitz Optimization

One of the main advantages of using a radial basis function surrogate to approximate the underlying expensive objective function is that we can cheaply obtain derivatives of the surrogate of any order. In particular, this allows us to easily obtain the Hessian Lipschitz constant over any ball in $\mathbb{R}^n$ and thus apply a Lipschitz based branch and bound algorithm. Note that this is an entirely new approach, distinct from the canonical Lipschitz branch and bound algorithm (see Section 5.3 of Pardalos et al., 1995).

6.5.1 Calculating the Lipschitz constant

We start by introducing tensors. A *third order tensor* T is a generalisation of a matrix to three indices, that is to say a 3-dimensional array. As with matrices $T_{i,j,k}$ denotes the i,j,k-th component (i.e. element in the array) of the tensor T. Recall that the matrix Frobenius norm can be defined as

$$\|A\|_F^2 := \sum_{i=1}^{m}\sum_{j=1}^{n} A_{i,j}^2$$

and this can be extended to a third order tensor T as

$$\|T\|_F^2 := \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{o} T_{i,j,k}^2.$$

Similarly, we can define the induced ℓ_2-norm for tensors by

$$\|T\|_2 := \max_{\|x\|_2=1} \|Tx\|_2$$

where $\|Tx\|_2$ denotes the usual induced matrix norm. We are now in a position to prove the following Lemma.

Lemma 6.16. *Let T be a third order tensor. Then $\|T\|_2 \leq \|T\|_F$.*

Proof: We have that

$$\begin{aligned}
\|T\|_2^2 &= \max_{\|x\|_2=1} \|Tx\|_2^2 \leq \max_{\|x\|_2=1} \|Tx\|_F^2 \quad \text{as } \|A\|_2 \leq \|A\|_F \text{ for matrices} \\
&= \max_{\|x\|_2=1} \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{k=1}^{o} T_{i,j,k}x_k\right)^2 \\
&= \max_{\|x\|_2=1} \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{k=1}^{o} (a_{i,j})_k x_k\right)^2 \text{ where the vector } a_{i,j} \text{ is s.t. } (a_{i,j})_k = T_{i,j,k} \\
&\leq \max_{\|x\|_2=1} \sum_{i=1}^{m}\sum_{j=1}^{n} \|a_{i,j}\|_2^2 \|x\|_2^2 \quad \text{by the Cauchy-Schwarz inequality} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} \|a_{i,j}\|_2^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{o} T_{i,j,k}^2 = \|T\|_F^2
\end{aligned}$$

□

Let $T(x) := \nabla_{xxx} l(x)$ denote the third order derivative tensor of the cost function $l(x)$. We have from Taylor's theorem that for any $x, y \in \mathcal{O}$

$$\|H(x) - H(y)\|_2 \leq \left\| \int_0^1 T\left(y + \tau(x-y)\right)(x-y) d\tau \right\|_2 \leq \max_{0 \leq \tau \leq 1} \|T\left(y + \tau(x-y)\right)\|_2 \|x - y\|_2$$

where $\|T(\cdot)\|_2$ denotes the tensor ℓ_2-norm defined above. Thus the Hessian $H(x) := \nabla_{xx} l(x)$ is Lipschitz continuous on a ball $\mathcal{O} \subset \mathbb{R}^n$ if there exists a ℓ_2-norm Lipschitz constant $L_H(\mathcal{O}) > 0$ such that for all $x \in \mathcal{O}$

$$\|T(x)\|_2 \leq L_H(\mathcal{O}).$$

It suffices to find an upper bound $\tau(\mathcal{O})$ on $T(x)$ over $\mathcal{O}$ and we can then use Lemma 6.16 to calculate an upper bound $L_H(\mathcal{O})$ on the optimal Hessian Lipschitz constant as

$$\|T(x)\|_2 \leq \|\tau(\mathcal{O})\|_2 \leq \|\tau(\mathcal{O})\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{o} \tau_{i,j,k}^2 \right)^{1/2} = L_H(\mathcal{O}).$$

Thus it remains to determine the upper bound $\tau(\mathcal{O})$ for the various cost functions given in Section 5.2. We will describe this in detail for the surrogate minimum cost function $l(x) = s(x)$. Recall from Chapter 3 that the surrogate $s(x)$ has the form

$$s(x) = \sum_{k=1}^{M} \mu_k \pi_k(x) + \sum_{j=1}^{N} \lambda_j \varphi(\|x - x_j\|_W)$$

with associated third order derivative tensor $T(x)$ given by

$$T(x) = \sum_{k=1}^{M} \mu_k \nabla_{xxx} \pi_k(x) + \sum_{j=1}^{N} \lambda_j \nabla_{xxx} \varphi(\|x - x_j\|_W).$$

To calculate the upper bound $\tau(\mathcal{O})$, it therefore suffices to calculate upper and lower bounds on the tensors $\nabla_{xxx}\pi_k(x)$ and $\nabla_{xxx}\varphi(\|x - x_j\|_W)$ over $\mathcal{O}$ depending on the signs of the coefficients λ_j, μ_k. For example, for the cubic spline RBF $\varphi(r) = r^3$ we have $\nabla_{xxx}\pi_k(x) = 0$ as the polynomial term is linear and

$$\left(\nabla_{xxx}\varphi(\|x - x_j\|_W)\right)_{a,b,c} = \begin{cases} \dfrac{-3w_a^6(x_a - x_{j_a})^3}{\|x - x_j\|_W^3} + \dfrac{9w_a^2(x_a - x_{j_a})}{\|x - x_j\|_W} & \text{if } a = b = c \\ \dfrac{-3w_c^2(x_c - x_{j_c})w_a^4(x_a - x_{j_a})^2}{\|x - x_j\|_W^3} + \dfrac{3w_c^2(x_c - x_{j_c})}{\|x - x_j\|_W} & \text{if e.g. } a = b \neq c \\ \dfrac{-3w_a^2(x_a - x_{j_a})w_b^2(x_b - x_{j_b})w_c^2(x_c - x_{j_c})}{\|x - x_j\|_W^3} & \text{otherwise.} \end{cases}$$

It is trivial to find upper and lower bounds on $w_a^2(x_a - x_{j_a})/\|x - x_j\|_W$ over the smallest box containing $\mathcal{O}$ which we can substitute into the above to obtain bounds on $\nabla_{xxx}\varphi(\|x - x_j\|_W)$. A similar approach can be used for the other basis functions with interval arithmetic techniques for higher order polynomial terms. Although it is in theory possible to apply this approach to other cost functions, it is quite involved as $\nabla_{xxx} l(x)$ will in general not be monotonic in $s(x)$ and $e(x)$.

6.5.2 Description of the Algorithm

We present an extension of the canonical branch and bound algorithm from Section 6.1 with bounds inspired by the trust region subproblem (see Chapter 7 of Conn, Gould and Toint, 2000). In this section we assume $\mathcal{D} \subset \mathbb{R}^n$ is a convex set and the cost function $l\colon \mathbb{R}^n \to \mathbb{R}$ is twice differentiable and Lipschitz continuous with Lipschitz continuous gradient $g(x) := \nabla_x l(x)$ and Hessian $H(x)$. In fact, we only need Lipschitz continuity of l, g and H over a sufficiently large region containing $\mathcal{D}$ but in the interests of keeping the proof simple we will assume this is $\mathbb{R}^n$. Let $\|\cdot\|$ denote the ℓ_2-norm and suppose there exist constants $L > 0$ and $L_g > 0$ such that

$$\|g(x)\| \leq L \text{ and } \|H(x)\| \leq L_g \tag{6.5.1}$$

for all $x \in \mathbb{R}^n$. Then L is an ℓ_2-norm Lipschitz constant for $l(x)$ and satisfies for all $x, y \in \mathbb{R}^n$

$$|l(x) - l(y)| \leq L\|x - y\| \tag{6.5.2}$$

and L_g is a gradient Lipschitz constant for $l(x)$ over $\mathbb{R}^n$. Similarly, let L_H denote the Hessian Lipschitz constant of $l(x)$ over $\mathbb{R}^n$. Note that these are all global Lipschitz constants. In this subsection, let $\mathcal{O} \subset \mathbb{R}^n$ denote the n-dimensional closed ball of radius $r(\mathcal{O}) > 0$ centred at some $x_\mathcal{O} \in \mathbb{R}^n$

$$\mathcal{O} = \{x \in \mathbb{R}^n : \|x - x_\mathcal{O}\| \leq r(\mathcal{O})\}.$$

Furthermore, let $L_H(\mathcal{O})$ denote the local Hessian Lipschitz constant for $l(x)$ over the ball $\mathcal{O}$ as calculated in the previous subsection. Define the upper and lower cubic bounding functions $m^{\pm}_\mathcal{O}\colon \mathbb{R}^n \to \mathbb{R}$ as (see Nesterov and Polyak, 2006)

$$m^{\pm}_\mathcal{O}(x) = l(x_\mathcal{O}) + (x - x_\mathcal{O})^T g(x_\mathcal{O}) + \frac{1}{2}(x - x_\mathcal{O})^T H(x_\mathcal{O})(x - x_\mathcal{O}) \pm \frac{L_H(\mathcal{O})}{6}\|x - x_\mathcal{O}\|^3.$$

Note that the upper and lower bounding functions over- and under-estimate $l(x)$ over $\mathcal{O}$. To see this, observe that we have from Taylor's theorem that for all $x \in \mathcal{O}$

$$\begin{aligned} l(x) = l(x_\mathcal{O}) &+ (x - x_\mathcal{O})^T g(x_\mathcal{O}) + \frac{1}{2}(x - x_\mathcal{O})^T H(x_\mathcal{O})(x - x_\mathcal{O}) \\ &+ \int_0^1 (1 - \tau)(x - x_\mathcal{O})^T \left[H\left(x_\mathcal{O} + \tau(x - x_\mathcal{O})\right) - H(x_\mathcal{O})\right](x - x_\mathcal{O})d\tau. \end{aligned} \tag{6.5.3}$$

Now, taking the absolute value of the integral gives for all $x \in \mathcal{O}$

$$\begin{aligned} &\left|\int_0^1 (1 - \tau)(x - x_\mathcal{O})^T \left[H\left(x_\mathcal{O} + \tau(x - x_\mathcal{O})\right) - H(x_\mathcal{O})\right](x - x_\mathcal{O})d\tau\right| \\ &\leq \int_0^1 (1 - \tau)\|H\left(x_\mathcal{O} + \tau(x - x_\mathcal{O})\right) - H(x_\mathcal{O})\|\|x - x_\mathcal{O}\|^2 d\tau \\ &\leq \int_0^1 (1 - \tau)L_H(\mathcal{O})\|\tau(x - x_\mathcal{O})\|\|x - x_\mathcal{O}\|^2 d\tau \qquad \text{as } H(\cdot) \text{ Lipschitz continuous} \\ &= \int_0^1 (1 - \tau)\tau d\tau\, L_H(\mathcal{O})\|x - x_\mathcal{O}\|^3 \end{aligned}$$

$$= \frac{L_H(\mathcal{O})}{6}\|x - x_{\mathcal{O}}\|^3.$$

We therefore have that for $x \in \mathcal{O}$ the integral can be bounded above and below by

$$\pm\frac{L_H(\mathcal{O})}{6}\|x - x_{\mathcal{O}}\|^3$$

which when combined with (6.5.3) gives for all $x \in \mathcal{O}$

$$m^-_{\mathcal{O}}(x) \leq l(x) \leq m^+_{\mathcal{O}}(x). \tag{6.5.4}$$

Define $\alpha(\mathcal{O})$ and $\beta(\mathcal{O})$ to be the lower and upper bounds for the global minimum of $l(x)$ over a ball $\mathcal{O} \subset \mathbb{R}^n$ as follows:

$$\alpha(\mathcal{O}) = \min_{x\in\mathcal{O}} m^-_{\mathcal{O}}(x) \tag{6.5.5}$$

and

$$\beta(\mathcal{O}) = l(x^+_{\mathcal{O}}) \tag{6.5.6}$$

where $x^+_{\mathcal{O}}$ is a feasible point in $\mathcal{O}$. Note that infeasible balls, i.e. balls which lie entirely outside of $\mathcal{D}$, are discarded by the algorithm (see step 1c) and the initial ball $\mathcal{O}_0$ contains $\mathcal{D}$. We discuss how $x^+_{\mathcal{O}}$ is calculated below. The algorithm proceeds as follows cf. the algorithm in Section 6.2:

Branch and Bound Algorithm for Lipschitz Optimization

0. Initialisation:

- *a) Set $k = 0$.*
- *b) Let $\mathcal{O}_0$ be a ball with centre $x_{\mathcal{O}} \in \mathcal{D}$ of sufficiently large radius to cover $\mathcal{D}$.*
- *c) Let $\mathcal{L}_0 = \{\mathcal{O}_0\}$ be the initial list of balls.*
- *d) Let $U_0 = \beta(\mathcal{O}_0)$ be the initial upper bound for $\min_{x\in\mathcal{D}} l(x)$.*
- *e) Let $L_0 = \alpha(\mathcal{O}_0)$ be the initial lower bound for $\min_{x\in\mathcal{D}} l(x)$.*

1. While $U_k - L_k > \varepsilon$, repeat the following procedure:

- *a) Remove from $\mathcal{L}_k$ balls $\mathcal{O} \in \mathcal{L}_k$ such that $\alpha(\mathcal{O}) > U_k$.*
- *b) Choose $\mathcal{O} \in \mathcal{L}_k$ such that $\alpha(\mathcal{O}) = L_k$.*
- *c) Split $\mathcal{O}$ into 3^n overlapping sub-balls $\mathcal{O}_1, \ldots, \mathcal{O}_{3^n}$ according to our splitting rule and discard any sub-balls which lie entirely outside of $\mathcal{D}$. Let $\mathcal{R}_k$ denote the list of remaining sub-balls and let $\mathcal{L}_{k+1} := (\mathcal{L}_k \setminus \{\mathcal{O}\}) \cup \mathcal{R}_k$.*
- *d) Set $U_{k+1} := \min_{\mathcal{O}\in\mathcal{L}_{k+1}} \beta(\mathcal{O})$.*
- *e) Set $L_{k+1} := \min_{\mathcal{O}\in\mathcal{L}_{k+1}} \alpha(\mathcal{O})$.*
- *f) Set $k = k + 1$.*

2. Return U_k as the estimate of the global minimum of $l(x)$ over $\mathcal{D}$.

Discarding Balls and Feasible Points

The algorithm discards balls $\mathcal{O}$ which lie entirely outside of $\mathcal{D}$. If $\mathcal{D}$ is a convex set this is easy to check and we do this as follows: Solving the convex QP

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \|x - x_{\mathcal{O}}\|^2 \\ &\text{s.t. } x \in \mathcal{D} \end{aligned}$$

provides a feasible minimiser $x_{\mathcal{O}}^+$ if the minimum is smaller than $r(\mathcal{O})^2$. This is how we calculate $x_{\mathcal{O}}^+ := \arg\min_{x\in\mathcal{D}}\|x - x_{\mathcal{O}}\|^2$ for (6.5.6). Moreover, if the minimum of the convex QP is larger than $r(\mathcal{O})^2$ we know that the ball $\mathcal{O}$ lies entirely outside of $\mathcal{D}$ and can be discarded.

Splitting Rule

We split a ball $\mathcal{O} \subset \mathbb{R}^n$ in step 1c of the above algorithm as follows. Let $\mathcal{O}$ have centre $x_{\mathcal{O}}$ and radius $r(\mathcal{O})$. Split $\mathcal{O}$ into 3^n sub-balls of radius $r(\mathcal{O})/2$ centred at the vertices of a hypercubic tessellation around $x_{\mathcal{O}}$ of edge length $r(\mathcal{O})/\sqrt{n}$. Formally, construct 3^n sub-balls $\mathcal{O}_1, \ldots, \mathcal{O}_{3^n}$ all of radius $r(\mathcal{O})/2$ centred at

$$x_{\mathcal{O}_i} = x_{\mathcal{O}} + \rho_i^n\left(\frac{-r(\mathcal{O})}{\sqrt{n}}, 0, \frac{r(\mathcal{O})}{\sqrt{n}}\right)$$

for $i = 1, \ldots, 3^n$. Here $\rho_i^n(s_1, s_2, s_3)$ is a vector in $\mathbb{R}^n$ whose elements are the i-th permutation of s_1, s_2, s_3 taken n at a time with repetition. We illustrate this for the case $n = 2$ in Figure 6.9. Note that the choice of centres and radii of the sub-balls ensures that they cover the original ball $\mathcal{O}$. Furthermore, this means that at any iteration of the above algorithm we always have a covering of closed balls of the convex set $\mathcal{D}$.

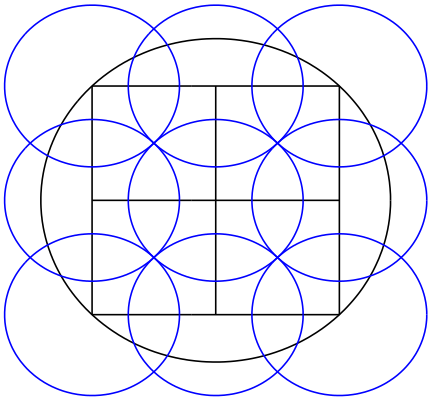

Figure 6.9: *An illustration of our splitting rule in two dimensions. The black circle is split into nine blue circles of half radius centred at the vertices of the square tessellation.*

6.5.3 Proof of Convergence

In this section we will prove that, under suitable assumptions, the above algorithm converges in a finite number of iterations to within a tolerance $\varepsilon > 0$ of the global minimum of $l(x)$. Our proof is based on the convergence proof of the canonical bound constrained branch and bound algorithm in Balakrishnan et al. (1991). First we state and prove a series of Lemmata before giving the main convergence theorem.

Lemma 6.17. *The bounds α and β given above in* (6.5.5) *and* (6.5.6) *respectively satisfy*

(C1) $\alpha(\mathcal{O}) \leq \min_{x\in\mathcal{O}} l(x) \leq \beta(\mathcal{O}) \quad \forall \mathcal{O} \subset \mathbb{R}^n$

(C2) $\forall \varepsilon > 0\ \exists \delta > 0$ *s.t.* $\forall \mathcal{O} \subset \mathbb{R}^n,\ r(\mathcal{O}) \leq \delta \implies \beta(\mathcal{O}) - \alpha(\mathcal{O}) \leq \varepsilon$

Proof: (C1) Recall that $\alpha(\mathcal{O}) = \min_{x\in\mathcal{O}} m^-_{\mathcal{O}}(x)$ and $\beta(\mathcal{O}) = l(x^+_{\mathcal{O}})$. From (6.5.4) we have that

$$\min_{x\in\mathcal{O}} m^-_{\mathcal{O}}(x) \leq \min_{x\in\mathcal{O}} l(x) \tag{6.5.7}$$

and clearly

$$\min_{x\in\mathcal{O}} l(x) \leq l(x^+_{\mathcal{O}}).$$

Thus we see that the bounds satisfy condition (C1).

(C2) Let $\varepsilon > 0$ be arbitrary. For clarity of exposition define for all $\mathcal{O} \subset \mathbb{R}^n$,

$$x^-_{\mathcal{O}} := \arg\min_{x\in\mathcal{O}} m^-_{\mathcal{O}}(x).$$

Note that $r(\mathcal{O}) \leq \delta$ means that

$$\|x^{\pm}_{\mathcal{O}} - x_{\mathcal{O}}\| \leq \delta. \tag{6.5.8}$$

Consider

$$\begin{aligned}
\beta(\mathcal{O}) - \alpha(\mathcal{O}) &= |l(x^+_{\mathcal{O}}) - m^-_{\mathcal{O}}(x^-_{\mathcal{O}})| \\
&\leq |l(x^+_{\mathcal{O}}) - l(x_{\mathcal{O}})| + \|x^-_{\mathcal{O}} - x_{\mathcal{O}}\|\|g(x_{\mathcal{O}})\| \\
&\quad + \frac{1}{2}\|x^-_{\mathcal{O}} - x_{\mathcal{O}}\|\|H(x_{\mathcal{O}})\|\|x^-_{\mathcal{O}} - x_{\mathcal{O}}\| + \frac{L_H}{6}\|x^-_{\mathcal{O}} - x_{\mathcal{O}}\|^3 \\
&\leq L\|x^+_{\mathcal{O}} - x_{\mathcal{O}}\| + \|x^-_{\mathcal{O}} - x_{\mathcal{O}}\|\|g(x_{\mathcal{O}})\| \\
&\quad + \frac{1}{2}\|x^-_{\mathcal{O}} - x_{\mathcal{O}}\|\|H(x_{\mathcal{O}})\|\|x^-_{\mathcal{O}} - x_{\mathcal{O}}\| + \frac{L_H}{6}\|x^-_{\mathcal{O}} - x_{\mathcal{O}}\|^3 \\
&\leq L\delta + \|g(x_{\mathcal{O}})\|\delta + \frac{1}{2}\|H(x_{\mathcal{O}})\|\delta^2 + \frac{L_H}{6}\delta^3 \\
&\leq 2L\delta + \frac{L_g}{2}\delta^2 + \frac{L_H}{6}\delta^3
\end{aligned}$$

where the first inequality follows directly from the triangle and Cauchy-Schwarz inequalities, the second from (6.5.2), the third from (6.5.8) and the fourth from (6.5.1). It therefore suffices to choose δ such that

$$2L\delta + \frac{L_g}{2}\delta^2 + \frac{L_H}{6}\delta^3 \leq \varepsilon$$

so for example, assuming equality above one can let δ be the single positive real root of the resulting cubic polynomial, which proves (C2).

□

Lemma 6.18. *The above algorithm eventually creates a ball of arbitrarily small radius. Formally, we have for $k \in \mathbb{N}$ that*

$$\min_{\mathcal{O}\in\mathcal{L}_k} r(\mathcal{O}) \leq \frac{r(\mathcal{O}_0)}{2^{(\log_{3^n} k)/2}}$$

and thus for any $\delta > 0$ there exists $K \in \mathbb{N}$ such that

$$\min_{\mathcal{O}\in\mathcal{L}_K} r(\mathcal{O}) \leq \delta.$$

Proof: Firstly recall that our splitting rule splits each ball into 3^n sub-balls. We start at iteration $k = 0$ with our initial covering ball $\mathcal{O}_0$ of radius $r(\mathcal{O}_0)$. We split $\mathcal{O}_0$ into 3^n sub-balls of radius $r(\mathcal{O}_0)/2$ at iteration $k = 1$. Assuming a worst case scenario, each of these 3^n sub-balls has to be split into 3^n subsub-balls of radius $r(\mathcal{O}_0)/4$ before we can consider any of the subsub-balls for splitting. Following this argument through inductively, we deduce that for $k \in \mathbb{N}$ it takes at worst

$$k = \sum_{j=1}^{m} (3^n)^{j-1} \tag{6.5.9}$$

iterations to reduce the radius of the smallest ball in the covering to less than or equal to

$$\frac{r(\mathcal{O}_0)}{2^m}. \tag{6.5.10}$$

We can bound (6.5.9) by

$$k \leq m(3^n)^{m-1} \leq (3^n)^{2m}$$

which when combined with (6.5.10) gives the required bound. The second part of the Lemma then follows trivially. □

Theorem 6.19. *The proposed algorithm converges in a finite number of iterations to within a tolerance $\varepsilon > 0$ of the global minimum of $l(x)$ over $\mathcal{D}$. Formally, for any $\varepsilon > 0$ there exists $M_\varepsilon \in \mathbb{N}$ such that*

$$U_{M_\varepsilon} - L_{M_\varepsilon} \leq \varepsilon$$

and U_{M_ε} is within a tolerance ε of the global minimum of $l(x)$ over $\mathcal{D}$.

Proof: Let $\mathcal{O}_k := \arg\min_{\mathcal{O}\in\mathcal{L}_k} r(\mathcal{O})$ and let $\mathcal{A}_k \in \mathcal{L}_{M_k}$ be the ball which for some $M_k < k$ we split to obtain $\mathcal{O}_k$. Let $\varepsilon > 0$ be arbitrary. Then there exists $\delta > 0$ such that for any $\mathcal{O} \subset \mathbb{R}^n$

$$r(\mathcal{O}) \leq 2\delta \implies \beta(\mathcal{O}) - \alpha(\mathcal{O}) \leq \varepsilon \tag{6.5.11}$$

by condition (C2) of Lemma 6.17. Choose $K \in \mathbb{N}$ sufficiently large such that

$$r(\mathcal{O}_K) \leq \delta$$

which is possible by Lemma 6.18. Then $\mathcal{A}_K$ must have $r(\mathcal{A}_K) \leq 2\delta$ as we split it to obtain $\mathcal{O}_k$ and thus from (6.5.11) we have that

$$\beta(\mathcal{A}_K) - \alpha(\mathcal{A}_K) \leq \varepsilon. \tag{6.5.12}$$

Now, as $\mathcal{A}_K$ was split at iteration M_K, it must have satisfied $\alpha(\mathcal{A}_K) = L_{M_K}$. Hence we get that

$$U_{M_K} - L_{M_K} \leq \beta(\mathcal{A}_K) - L_{M_K} \leq \varepsilon \tag{6.5.13}$$

since $U_{M_K} \leq \beta(\mathcal{A}_K)$ by definition and using (6.5.12). We therefore have an upper bound M_K on the number of branch and bound iterations.

It remains to show that U_{M_K} is within a tolerance ε of the global minimum of $l(x)$ over $\mathcal{D}$. Assume condition (C1) of Lemma 6.17 holds and suppose that the global minimum l^* of $l(x)$ over $\mathcal{D}$ is attained at $x^* \in \mathcal{D}$. First, we show that x^* is contained in a ball in $\mathcal{L}_{M_K}$. To see this, observe that for all $k \in \mathbb{N}$, $\mathcal{L}_k$ is a partition of the bounding ball $\mathcal{O}_0$ with balls which cannot possibly contain x^* removed, that is to say balls $\mathcal{O}$ which have lower bound

$$\alpha(\mathcal{O}) > U_k$$

i.e. $\alpha(\mathcal{O}) > l(x^+_{\mathcal{O}})$ for a feasible point $x^+_{\mathcal{O}} \in \mathcal{D}$. As x^* is contained in a ball in $\mathcal{L}_{M_K}$ it follows that $L_{M_K} \leq l^*$ and thus

$$U_{M_K} - l^* \leq U_{M_K} - L_{M_K} \leq \varepsilon$$

by (6.5.13). □

6.5.4 A Faster Heuristic Algorithm

All of the algorithms in this Chapter are aimed primarily at solving the ancillary optimization problem which occurs within the sequential decision theoretic optimization framework in Sections 5.2 and 5.4. In view of this, it is not always necessary or desirable to find the global minimum to a high degree of accuracy (see Mockus, 1994). With this in mind we present a faster heuristic version of the Lipschitz based branch and bound algorithm which has no theoretical convergence guarantees but still exhibits reasonable performance.

The main drawback with regards to performance of the existing algorithm is the splitting of each ball into 3^n sub-balls since the number of sub-balls grows rapidly as n (the dimension of the problem) increases. Rather than using overlapping balls, for the heuristic version of the algorithm we split each ball into a dense lattice of non-overlapping sub-balls. The maximum number of balls one can pack around a central ball without overlap is given by the kissing number κ (Table 6.1). Optimal kissing numbers and the corresponding lattices we use which give rise to them are known up to 9 dimensions (see Conway and Sloane, 1999, for details). Running the algorithm with this splitting rule means that each ball is only split into $\kappa + 1$ sub-balls, considerably less than 3^n. The disadvantage is that it leaves holes in the domain we are trying to optimize over and so convergence to the global optimum is not guaranteed. However, by running a local solver from the global minimum proposed by the algorithm,

n	κ	3^n
1	2	3
2	6	9
3	12	27
4	24	81
5	40	243
6	72	729
7	126	2187
8	240	6561
9	272	19683

Table 6.1: *The optimal kissing number κ compared against 3^n for the first 9 dimensions.*

we will always find a local minimum which is often a good candidate for being the global optimum (and which can be used as an upper bound in the original slower algorithm).

6.5.5 Computing the Bounds

We have mentioned earlier in Subsection 6.5.2 that we use upper and lower bounds based on globally minimising the cubic bounding functions $m^{\pm}_{\mathcal{O}}(x)$ over balls $\mathcal{O} \subset \mathbb{R}^n$, due to Nesterov and Polyak (2006). In this subsection we will show how we can efficiently globally minimise $m^{\pm}_{\mathcal{O}}(x)$ over any closed ball $\mathcal{O}$ centred at $x_{\mathcal{O}}$, i.e.

$$\text{minimise } m^{\pm}_{\mathcal{O}}(x) = l(x_{\mathcal{O}}) + (x - x_{\mathcal{O}})^T g(x_{\mathcal{O}}) + \frac{1}{2}(x - x_{\mathcal{O}})^T H(x_{\mathcal{O}})(x - x_{\mathcal{O}}) \pm \frac{L_H(\mathcal{O})}{6}\|x - x_{\mathcal{O}}\|^3$$
$$\text{subject to } \|x - x_{\mathcal{O}}\| \leq \Delta$$

for some $\Delta > 0$. For clarity of exposition, we rewrite the above minimisation problem in the equivalent form

$$\text{minimise } m^{\pm}(x) := f + x^T g + \frac{1}{2}x^T H x \pm \frac{\sigma}{3}\|x\|^3$$
$$\text{subject to } \|x\| \leq \Delta$$

where $f := l(x_{\mathcal{O}}), \sigma := L_H(\mathcal{O})/2$ and we have dropped the explicit dependence on $\mathcal{O}$ from the notation. It is clear that the global minimum of the above problem will occur either on the boundary or in the interior of the Δ-ball. We solve for these two cases in turn, starting with the case where the minimum lies on the boundary.

Finding a minimiser on the Δ-ball boundary

For the bounding functions $m^{\pm}(x)$ we have the following global optimality result (cf. Theorem 7.2.1 in Conn et al., 2000 and Theorem 3.1 in Cartis, Gould and Toint, 2009).

Theorem 6.20. *Any x^* is a global minimiser of $m^{\pm}(x)$ over $\mathbb{R}^n$ subject to $\|x\| = \Delta$ if and only if it satisfies the system of equations*

$$(H + (\lambda^* \pm \sigma\Delta)I)x^* = -g \tag{6.5.14}$$

where $H+(\lambda^*\pm\sigma\Delta)I$ is positive semidefinite for some Lagrange multiplier λ^* and $\|x^*\| = \Delta$. If $H + (\lambda^* \pm \sigma\Delta)I$ is positive definite, x^* is unique.

Proof: First we rewrite the constraint $\|x\| = \Delta$ as $\frac{1}{2}\|x\|^2 - \frac{1}{2}\Delta^2 = 0$. Now, let x^* be a global minimiser of $m^{\pm}(x)$ over $\mathbb{R}^n$ subject to the constraint. We have from the first order necessary optimality conditions (see Section 3.2.2 of Conn et al., 2000) that x^* satisfies

$$(H + (\lambda^* \pm \sigma\|x^*\|)I)x^* = -g. \tag{6.5.15}$$

where λ^* is the corresponding Lagrange multiplier. We have by assumption that $\|x^*\| = \Delta$ and substituting this into (6.5.15) gives the required system (6.5.14). Now, suppose u^* is a feasible point, i.e. that $\|u^*\| = \Delta$. We have that

$$\begin{aligned} m^{\pm}(u^*) - m^{\pm}(x^*) &= g^T(u^* - x^*) + \frac{1}{2}(u^*)^T H u^* - \frac{1}{2}(x^*)^T H x^* + \frac{\sigma}{3}\left(\|u^*\|^3 - \|x^*\|^3\right) \\ &= g^T(u^* - x^*) + \frac{1}{2}(u^*)^T H u^* - \frac{1}{2}(x^*)^T H x^* \end{aligned} \tag{6.5.16}$$

where the last equality follows from the fact that $\|x^*\| = \|u^*\| = \Delta$. But (6.5.15) gives that

$$g^T(u^* - x^*) = (x^* - u^*)^T H x^* + (\lambda^* \pm \sigma\Delta)(x^* - u^*)^T x^*. \tag{6.5.17}$$

Also, the fact that $\|x^*\| = \|u^*\| = \Delta$ implies that

$$(x^* - u^*)^T x^* = \frac{1}{2}(x^*)^T x^* + \frac{1}{2}(u^*)^T u^* - (u^*)^T x^* = \frac{1}{2}(u^* - x^*)^T(u^* - x^*). \tag{6.5.18}$$

Combining (6.5.16) with (6.5.17) and (6.5.18), we find that

$$\begin{aligned} m^{\pm}(u^*) - m^{\pm}(x^*) &= \frac{1}{2}(\lambda^* \pm \sigma\Delta)(u^* - x^*)^T(u^* - x^*) + \frac{1}{2}(u^*)^T H u^* \\ &\quad - \frac{1}{2}(x^*)^T H x^* + (x^*)^T H x^* - (u^*)^T H x^* \\ &= \frac{1}{2}(u^* - x^*)^T(H + (\lambda^* \pm \sigma\Delta)I)(u^* - x^*). \end{aligned} \tag{6.5.19}$$

We also have from the second order necessary optimality conditions (see Section 3.2.2 of Conn et al., 2000) that

$$H + (\lambda^* \pm \sigma\|x^*\|)I \pm \frac{\sigma}{\|x^*\|}x^*(x^*)^T$$

is positive semidefinite on the null-space of the constraint gradient x^*, i.e. that

$$v^T\left(H + (\lambda^* \pm \sigma\Delta)I \pm \frac{\sigma}{\Delta}x^*(x^*)^T\right)v \geq 0 \tag{6.5.20}$$

for all v for which $v^T x^* = 0$, where we have used the fact that $\|x^*\| = \Delta$. In this case it immediately follows from (6.5.20) that

$$v^T\left(H + (\lambda^* \pm \sigma\Delta)I\right)v \geq 0$$

for all v for which $v^T x^* = 0$. It thus remains to consider vectors v for which $v^T x^* \neq 0$. Since v and x^* are not orthogonal, the line $x^* + \alpha v$ intersects the constraint $\|x\| = \Delta$ at

two points, x^* and u^*. Let $v^* = u^* - x^*$ and note that v^* is parallel to v. As x^* is a global minimiser we have that $m^{\pm}(u^*) \geq m^{\pm}(x^*)$ and thus we have from (6.5.19) that

$$\begin{aligned} 0 \leq m^{\pm}(u^*) - m^{\pm}(x^*) &= \frac{1}{2}(u^* - x^*)^T (H + (\lambda^* \pm \sigma\Delta)I)(u^* - x^*) \\ &= \frac{1}{2}(v^*)^T (H + (\lambda^* \pm \sigma\Delta)I)v^* \end{aligned} \tag{6.5.21}$$

from which we deduce that

$$v^T \left(H + (\lambda^* \pm \sigma\Delta)I\right) v \geq 0$$

for all v for which $v^T x^* \neq 0$. In summary, we have shown that

$$v^T \left(H + (\lambda^* \pm \sigma\Delta)I\right) v \geq 0$$

for any vector, which is the same as saying that $H + (\lambda^* \pm \sigma\Delta)I$ must be positive semidefinite. Conversely, if $H + (\lambda^* \pm \sigma\Delta)I$ is positive definite, $\frac{1}{2}(u^* - x^*)^T (H + (\lambda^* \pm \sigma\Delta)I)(u^* - x^*) > 0$ for any $u^* \neq x^*$ and therefore (6.5.21) shows that $m^{\pm}(u^*) > m^{\pm}(x^*)$ whenever u^* is feasible. Thus x^* is the unique global minimiser. □

The global minimiser can be efficiently found by applying a safeguarded version of Newton's method as detailed in Section 2.1 of Gould, Robinson and Thorne (2010) to the scalar equation

$$\begin{aligned} \|x(\lambda)\| &= \Delta, \\ \text{where } (H + (\lambda \pm \sigma\Delta)I)x(\lambda) &= -g, \end{aligned}$$

and this is the approach we take (see Figure 6.10 below).

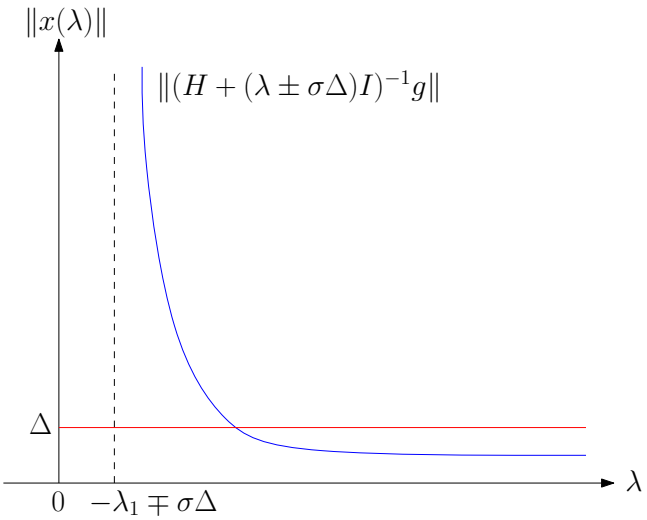

Figure 6.10: *Solutions to the system* (6.5.14) *are the intersections of the two curves.*

Finding a minimiser in the Δ-ball interior

For the upper bounding function $m^+(x)$ we have the following global optimality result.

Theorem 6.21 (Theorem 3.1 in Cartis et al., 2009)**.** *Any x^* is a global minimiser of $m^+(x)$ over $\mathbb{R}^n$ if and only if it satisfies the system of equations*

$$(H + \omega^* I)x^* = -g \tag{6.5.22}$$

where $\omega^* = \sigma\|x^*\|$ and $H + \omega^* I$ is positive semidefinite. If $H + \omega^* I$ is positive definite, x^* is unique.

Proof: See the proof of Theorem 3.1 in Cartis et al. (2009). □

As before, the global minimiser can be efficiently found by applying a safeguarded version of Newton's method (Algorithm 6.1 in Cartis et al., 2009) to the scalar equation

$$\|x(\omega)\| = \frac{\omega}{\sigma},$$
$$\text{where } (H + \omega I)x(\omega) = -g.$$

This approach is illustrated in Figure 6.11 below.

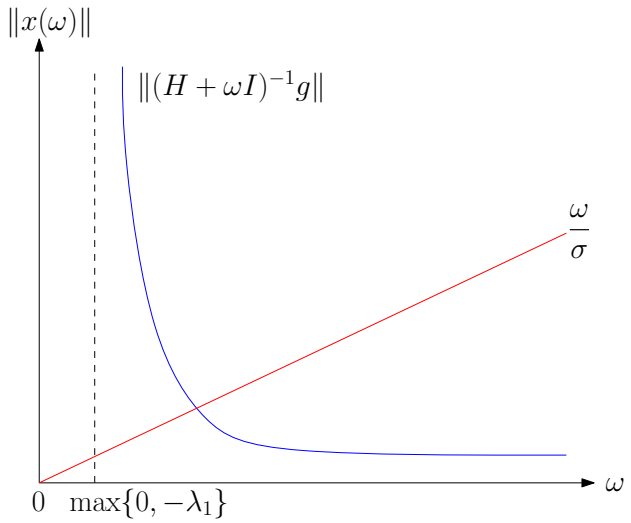

Figure 6.11: *Solutions to the system* (6.5.22) *for $m^+(x)$ are the intersections of the two curves.*

For the lower bounding function $m^-(x)$ we have the following weaker result:

Theorem 6.22. *The lower bounding function $m^-(x)$ can only have finite global minimisers if H is positive semidefinite. In this case, any x^* is a global minimiser of $m^-(x)$ over $\mathbb{R}^n$ if and only if it satisfies the system of equations*

$$(H + \omega^* I)x^* = -g \tag{6.5.23}$$

where $\omega^* = -\sigma\|x^*\|$ and $H + \omega^* I$ is positive semidefinite.

Proof: For the first part, suppose H is not positive semidefinite and consider $x = \alpha u$ for any eigenvector u of H corresponding to a negative eigenvalue. Then clearly $m^-(x) \to -\infty$ as $\alpha \to \infty$ and so the function $m^-(x)$ is unbounded below. The second part of the proof is analogous to the proof of the first part of Theorem 3.1 in Cartis et al. (2009). □

Note that in this case $\omega^* = -\sigma\|x^*\|$ and so there can only be a solution for $\omega \leq 0$. Assuming H is positive semidefinite, let λ_1 denote the smallest eigenvalue of H and note that if $\lambda_1 = 0$ there can only be a trivial solution to the system (6.5.23) when $x = 0$ and $g = 0$. When $\lambda_1 > 0$ there will be at most two possible solutions to the system (6.5.23) which, once again, can be found using Newton's method with suitable starting points (i.e. Algorithm 6.1 in Cartis et al., 2009). Numerical results suggest that in this case the solution closest to zero is always the best local minimiser in the Δ-ball interior, but this remains to be proved. Figure 6.12 below illustrates this typical case when there are two possible solutions. Note that there may be no solutions and an example of this is the case where the straight line lies under the curve in Figure 6.12.

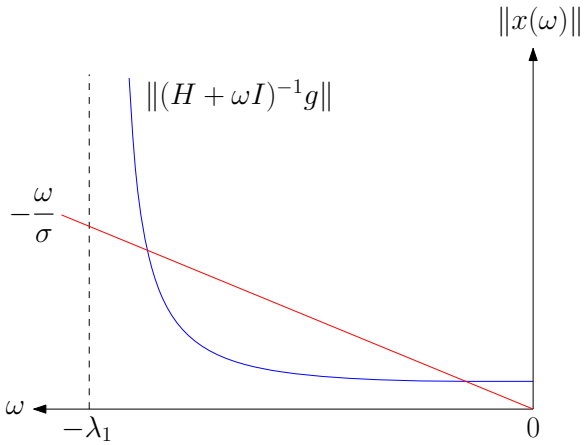

Figure 6.12: *Solutions to the system* (6.5.23) *for* $m^-(x)$ *are the intersections of the two curves.*

6.5.6 Numerical Example

Consider once more the problem of finding the global minimum of a cubic spline RBF surrogate $s(x, y)$ fitted to the Dixon-Szegő six hump camel back function at thirty scattered points in $[-2, 2] \times [-1.25, 1.25]$. This time we will use our branch and bound algorithm with overlapping balls and local Lipschitz constants from Subsection 6.5.2. The optimal solution as found in 244 iterations of step 1 of the algorithm with a tolerance of 4×10^{-6} is shown in Figure 6.13 below. This took about 25 seconds of cpu time for a Matlab implementation on an AMD Phenom II X3 705e processor machine (cf. the numerical example in Section 6.1).

Compare this with the heuristic version using a lattice of balls from Subsection 6.5.4. In this case, the optimal solution as found in 147 iterations of step 1 of the algorithm with a

tolerance of 6×10^{-6} is shown in Figure 6.14 below. This took about 12 seconds of cpu time, considerably less.

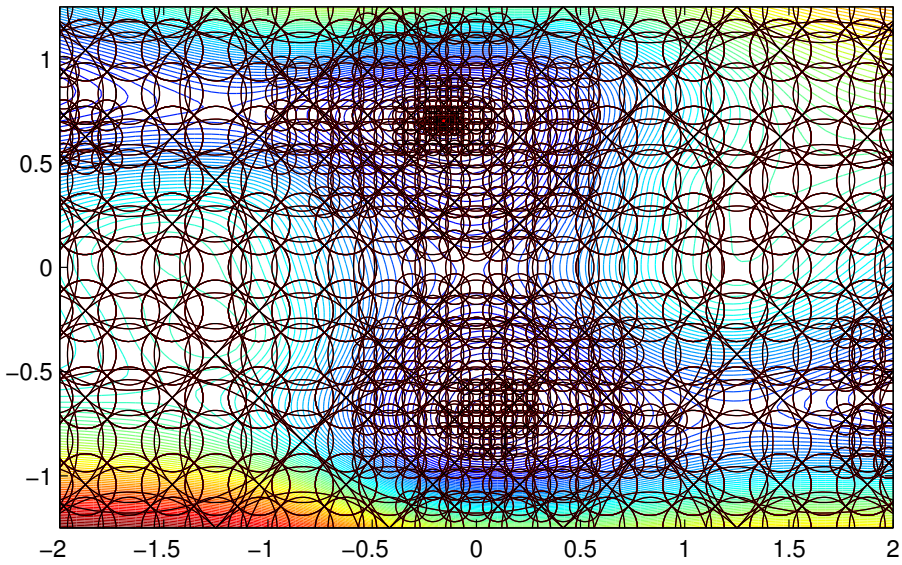

Figure 6.13: *Contours of the RBF surrogate $s(x, y)$ fitted to the camel function. The black circles denote the overlapping closed balls $\mathcal{O}$ used by the branch and bound algorithm. Note that they cluster at the global minimum of $s(x, y)$ which is denoted by a red circle.*

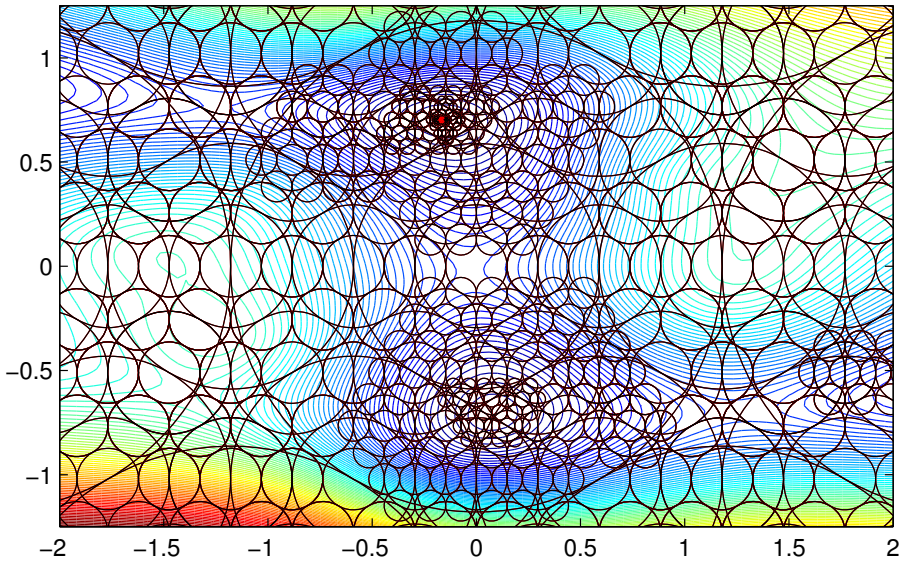

Figure 6.14: *Contours of the RBF surrogate $s(x, y)$ fitted to the camel function. The black circles denote the hexagonal lattice of balls $\mathcal{O}$ used by the branch and bound algorithm. Note that they cluster at the global minimum of $s(x, y)$ which is denoted by a red circle.*

Chapter 7

Integration

Suppose that we are interested in finding the integral of our expensive objective function $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ over some domain $\mathcal{D}$, that is to say

$$\int_{\mathcal{D}} f(x)dx.$$

Given our surrogate framework from Chapter 3, a natural approach would be to replace the objective function f with a surrogate approximation $s(x)$ as suggested by O'Hagan (1992). Further, following on from the one-stage lookahead framework in Chapter 5 we can sequentially improve the surrogate before integrating it in place of the objective function. We will show by means of examples that this proposed method is comparable to existing Monte Carlo approaches to integration of expensive functions in low dimensions. It is important to note that this approach is not suitable for high dimensional integration since the rate of convergence depends on the surrogate (see Section 3.6) and therefore on the dimension. Monte Carlo integration on the other hand converges at a rate of $O(N^{-1/2})$ independent of dimension (see Caflisch, 1998).

7.1 Outline of the Method

As we have a surrogate approximation $s(x)$ to the objective function $f(x)$, an obvious approach to obtain an approximation to the integral of $f(x)$ is to integrate the surrogate. However, the integral of the surrogate will only be a good approximation to the integral of the objective function if the surrogate is itself a good approximation to the objective function. This can be achieved by minimising the error $e(x)$ (or equivalently the variance $e^2(x)$) associated with the surrogate. Indeed, we have shown in Subsection 3.6.5 that for

$f \in \mathcal{N}_\varphi(\mathcal{D})$

$$|f(x) - s(x)| \leq \frac{\|f\|_{\mathcal{N}_\varphi(\mathcal{D})}}{\sigma} e(x)$$

and therefore the absolute integration error satisfies

$$\left| \int_{\mathcal{D}} f(x)dx - \int_{\mathcal{D}} s(x)dx \right| = \left| \int_{\mathcal{D}} f(x) - s(x)dx \right| \leq \int_{\mathcal{D}} |f(x) - s(x)|dx \leq \frac{\|f\|_{\mathcal{N}_\varphi(\mathcal{D})}}{\sigma} \int_{\mathcal{D}} e(x)dx.$$

This shows that (at least for $f \in \mathcal{N}_\varphi(\mathcal{D})$) including the maximiser of the error $e(x)$ in the surrogate would reduce the bound on the absolute integration error. This is because the peak in $e(x)$ associated with its maximum would collapse (as sampled points x have $e(x) = 0$) thus decreasing the area under $e(x)$ and hence its integral. With this motivation, we can sequentially improve the surrogate $s(x)$ before integration by applying our one-stage lookahead framework from Section 5.2 with the error $e(x)$ as a cost function.

A more natural approach is to minimise the variance of the integral itself, i.e. the variance of $\int_{\mathcal{D}} f(x)dx$, given in Section 4.2. However as it is difficult to obtain bounds for the variance of the integral and thus to optimise it with our branch and bound algorithms from Chapter 6 we do not consider it here and instead refer the interested reader to O'Hagan (1991).

Following our initial approach, we take the maximum error cost function and apply our one-stage lookahead framework to sequentially improve the surrogate. Once we have finished improving the surrogate we integrate it over $\mathcal{D}$. For this reason we restrict ourselves to radial basis functions and domains for which the surrogate is computationally inexpensive to integrate. Recall from Chapter 3 that the surrogate $s(x)$ takes the form

$$s(x) = \sum_{k=1}^{M} \mu_k \pi_k(x) + \sum_{j=1}^{N} \lambda_j \varphi(\|x - x_j\|_W).$$

Integrating the surrogate gives

$$\int_{\mathcal{D}} s(x)dx = \sum_{k=1}^{M} \mu_k \int_{\mathcal{D}} \pi_k(x)dx + \sum_{j=1}^{N} \lambda_j \int_{\mathcal{D}} \varphi(\|x - x_j\|_W)dx$$

and we evaluate the integrals for various domains below. We give illustrative numerical examples for two domains where analytical integration of the surrogate is possible.

For a given objective function $f(x)$ let I denote the exact integral of $f(x)$ over $\mathcal{D}$ and let I_N denote the integral of the surrogate $s(x)$ fitted to N points $x_1, \ldots, x_N$. Define the absolute error at step N by

$$|I - I_N|$$

and the relative error at step N by

$$\frac{|I - I_N|}{|I|}.$$

We compare our method against the standard Monte Carlo approach. Given N pseudo-random points $x_1, \ldots, x_N$ in $\mathcal{D}$ (see Section 4.1), the approximate Monte Carlo integral M_N of $f(x)$ over $\mathcal{D}$ is given by

$$M_N = \frac{\text{vol}(\mathcal{D})}{N} \sum_{i=1}^{N} f(x_i)$$

where $\mathrm{vol}(\mathcal{D})$ denotes the volume of $\mathcal{D}$. The absolute and relative errors for Monte Carlo are defined in exactly the same way.

7.2 Rectangular Domain

We consider the canonical example of the rectangular domain $\mathcal{D} = [l, u]^n$. Let π_k be the linear monomial basis in $\mathbb{R}^n$, i.e. $\pi_1(\xi) = 1, \pi_2(\xi) = \xi_1, \pi_3(\xi) = \xi_2$, etc. and let $\varphi(\cdot)$ be the Gaussian radial basis function, i.e. $\varphi(\|x - x_j\|_W) = \exp(-\|x - x_j\|_W^2)$. Then it is easy to show that

$$\int_{\mathcal{D}} \pi_k(x)dx = \begin{cases} \mathrm{vol}(\mathcal{D}) & \text{if } k = 1 \\ \frac{1}{2}(u_k - l_k)\mathrm{vol}(\mathcal{D}) & \text{otherwise} \end{cases}$$

where $\mathrm{vol}(\mathcal{D}) = (u_1 - l_1)(u_2 - l_2)\dots(u_n - l_n)$ is the volume of $\mathcal{D}$ and

$$\int_{\mathcal{D}} \varphi(\|x - x_j\|_W)dx = \frac{(-1)^n \pi^{n/2}}{2^n} \prod_{i=1}^{n} \frac{\mathrm{erf}(w_i(l_i - x_{j_i})) - \mathrm{erf}(w_i(u_i - x_{j_i}))}{w_i}$$

where $\mathrm{erf}(\cdot)$ denotes the error function.

As a numerical example, let $\mathcal{D} = [0, 1]^n$ for $n = 5$ and consider the test function (see Genz, 1984)

$$f(x) = \frac{1}{\left(1 + \frac{1}{n}\sum_{i=1}^{n} x_i\right)^{(0.3+n)}}$$

which has exact integral over $\mathcal{D}$ given by

$$I = \frac{1}{\prod_{k=0}^{n-1} \frac{k+0.3}{n}} \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} \frac{(-1)^{\sum_{k=1}^{n} i_k}}{\left(1 + \sum_{k=1}^{n} \frac{i_k}{n}\right)^{0.3}}.$$

We can now apply our integration method using iterative refinement from Section 7.1 and compare it against the standard Monte Carlo approach as well as the integral of the surrogate at random points (i.e. without iterative refinement) which we use as a control. In order to get an indication of which method is preferable, we run each algorithm ten times and average over the ten runs. For our integration algorithm, we start with $N = 25$ initial samples of f and iteratively refine the approximation for another 35 samples. The global optimization of the surrogate error $e(x)$ is performed using our C++ implementation of the canonical branch and bound algorithm from Section 6.1. The mean of the absolute error over ten runs of each algorithm is given in Figure 7.1 and the mean absolute error $\pm$ two standard deviations in Figure 7.2 overleaf. We compare our algorithm at each step N with an N point Monte Carlo approximation as well as the integral of a surrogate fitted to N random points. In this example we can see that integrating the surrogate both with and without iterative refinement gives a consistently better approximation to the integral than Monte Carlo. However, the extra computational cost required in the iterative refinement cannot be justified for this particular example as the performance is comparable to integrating the surrogate without refinement.

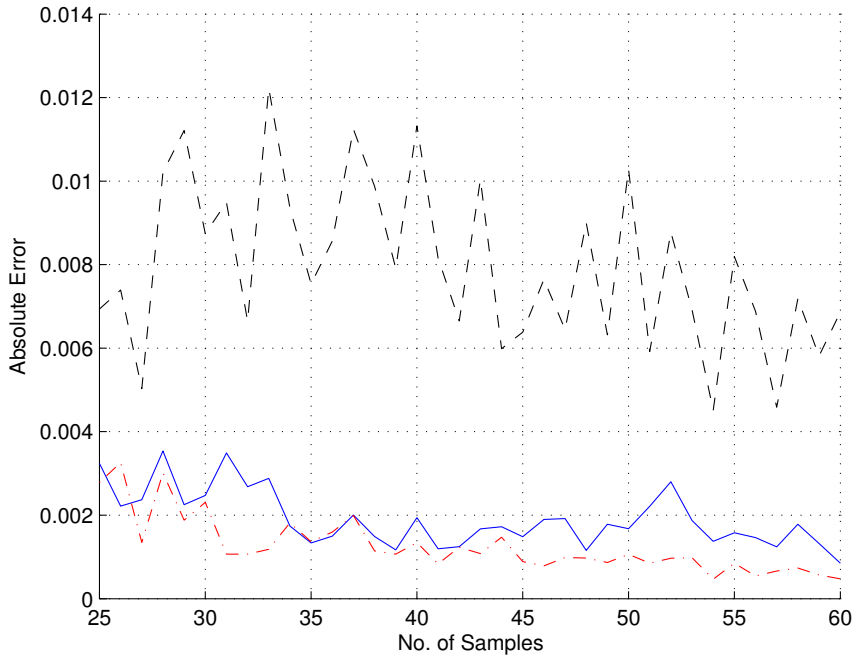

Figure 7.1: *The mean absolute error at each step (i.e. sample) of our integration algorithm (solid blue), Monte Carlo (dashed black) and the control (dash-dotted red).*

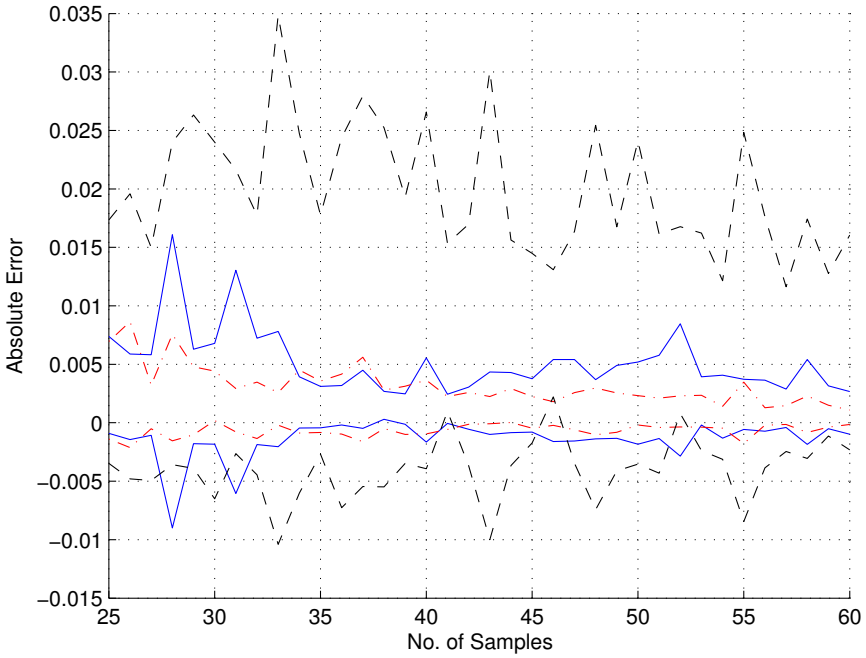

Figure 7.2: *The mean absolute error ± two standard deviations at each step (i.e. sample) of our integration algorithm (solid blue), Monte Carlo (dashed black) and the control (dash-dotted red).*

7.3 Spherical Surface

Let $\mathcal{D} = S^{n-1}$, the surface of the n-dimensional sphere. Then we can write

$$\varphi(\|x - x_j\|) = \varphi(\sqrt{2 - 2x^T x_j}) = \varphi(\sqrt{2 - 2t})$$

and view each RBF term as a function of t. It can be shown (see Hubbert and Baxter, 2001) that each RBF term has an expansion

$$\varphi(\sqrt{2 - 2t}) = \sum_{l=0}^{\infty} a_l P_l^{\lambda}(t) \tag{7.3.1}$$

on the sphere where P_l^{λ} are the Gegenbauer polynomials and the coefficients a_l are known. For example, for the odd spline RBF $\varphi(r) = r^p$ we have

$$a_l = (-1)^l \pi^{-0.5} 2^{2\lambda} (l + \lambda)\Gamma(\lambda) \frac{2^p \Gamma(p/2 + 1)\Gamma((p + 1)/2 + \lambda)}{\Gamma(p/2 + 1 - l)\Gamma(p/2 + l + 1 + 2\lambda)}$$

where $\Gamma(\cdot)$ denotes the Gamma function. Again, let π_k be the linear monomial basis in $\mathbb{R}^n$ and let $\varphi(\cdot)$ be a radial basis function. Then it can be shown that

$$\int_{\mathcal{D}} \pi_k(x)dx = \begin{cases} \text{vol}(\mathcal{D}) & \text{if } k = 1 \\ 0 & \text{otherwise} \end{cases}$$

where $\text{vol}(\mathcal{D}) = n\pi^{n/2}/\Gamma(n/2 + 1)$ is the surface area of the n-sphere and using (7.3.1)

$$\int_{\mathcal{D}} \varphi(\|x - x_j\|)dx = \sum_{l=0}^{\infty} a_l \int_{\mathcal{D}} P_l^{\lambda}(t)dx = a_0 \text{vol}(\mathcal{D}).$$

As an illustrative numerical example, let $\mathcal{D} = S^{n-1}$ for $n = 5$ and consider the test function

$$f(x) = \prod_{i=1}^{n} x_i^2$$

which has exact integral over $\mathcal{D}$ given by

$$I = \frac{2\prod_{i=1}^{n} \Gamma(\frac{3}{2})}{\Gamma\left(\frac{3n}{2}\right)}.$$

Further, let $\varphi(\|x - x_j\|) = \|x - x_j\|^3$, the cubic spline radial basis function. As before, we can apply our integration method using iterative refinement from Section 7.1 and compare it against the standard Monte Carlo approach as well as the integral of the surrogate at random points (i.e. without iterative refinement) which we use as a control. In order to get an indication of which method is preferable, we run each algorithm eight times and average over the eight runs. For our integration algorithm, we start with $N = 25$ initial samples of f and iteratively refine the approximation for another 35 samples. The global optimization of the surrogate error $e(x)$ is performed using a Matlab implementation of our branch and bound algorithm for S^{n-1} from Section 6.4. The mean of the relative error

over ten runs of each algorithm is given in Figure 7.3 and the mean relative error $\pm$ two standard deviations in Figure 7.4. As before, we compare our algorithm at each step N with an N point Monte Carlo approximation as well as the integral of a surrogate fitted to N random points. We can see that integrating the surrogate with iterative refinement gives a comparable approximation to the integral than Monte Carlo. Moreover, in this case iteratively refining the surrogate is much better than integrating it without performing any refinement.

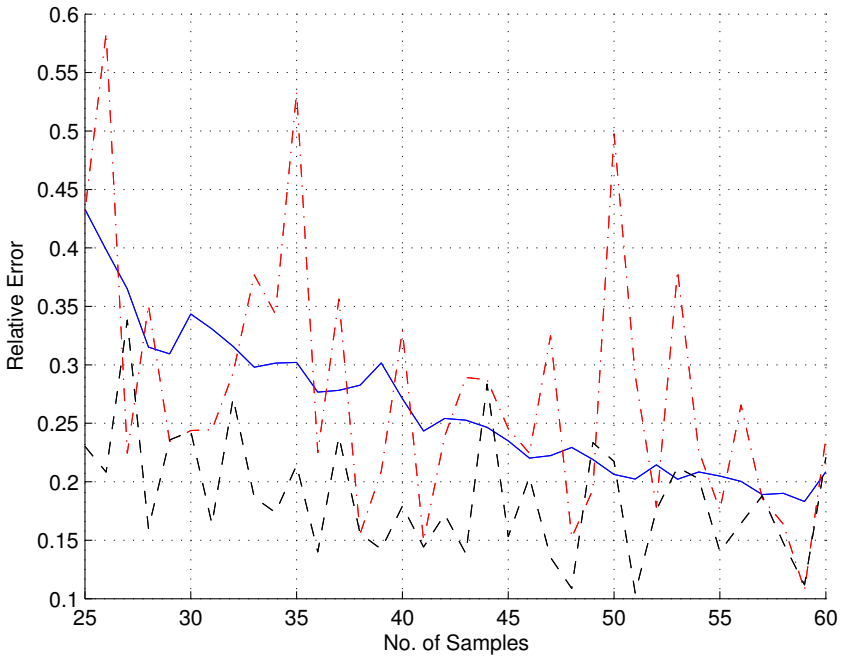

Figure 7.3: *The mean relative error at each step (i.e. sample) of our integration algorithm (solid blue), Monte Carlo (dashed black) and the control (dash-dotted red).*

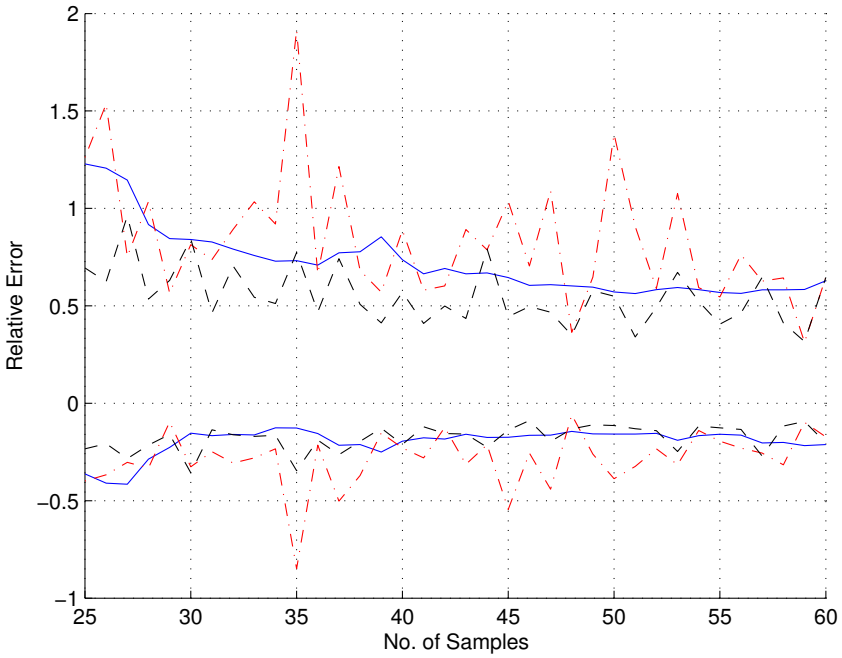

Figure 7.4: *The mean relative error ± two standard deviations at each step (i.e. sample) of our integration algorithm (solid blue), Monte Carlo (dashed black) and the control (dash-dotted red).*

Chapter 8

Numerical Examples

In this chapter we present some numerical examples of the one-stage lookahead global optimization method proposed in Section 5.2. We will first look at some of the advantages our surrogate framework from Chapter 3 and proposed branch and bound algorithms from Chapter 6 offer before looking at the performance of the method on some canonical examples. We then apply it to a model problem in reservoir simulation, the optimal well placement problem.

8.1 Canonical Examples

Let us first consider the advantages our surrogate framework from Chapter 3 offers over existing methods in the literature such as RBF interpolation without a weighted norm. From the perspective of scattered data approximation (see Wendland, 2005), RBF interpolation is performed using the ℓ_2-norm and this performs well when we can take a large number of samples of the objective function. However, this is not possible when our objective function is expensive and it is in such a setting, when we are limited to a small number of samples, that the anisotropy in a weighted RBF interpolant can be advantageous. We will use the cubic spline RBF throughout this chapter to illustrate our proposed surrogate framework as it has been shown to perform best among the low order spline basis functions (see Gutmann, 2001) and is numerically very robust, as we have shown in Section 3.6.

It is evident then, that there will be cases where using weighted RBF interpolation will yield better results than standard RBF interpolation. We illustrate our assertion by means of an example. Consider the Branin function in $\mathbb{R}^2$ (see Dixon and Szegő, 1978) on the interval $\mathcal{D} = [-5, 10] \times [0, 15]$. It has three global minima as one can see in Figure 8.1 overleaf. We construct RBF and weighted RBF surrogates to the Branin function at twenty maximin Latin hypercube sample points (see Section 4.2) and compare the accuracy of

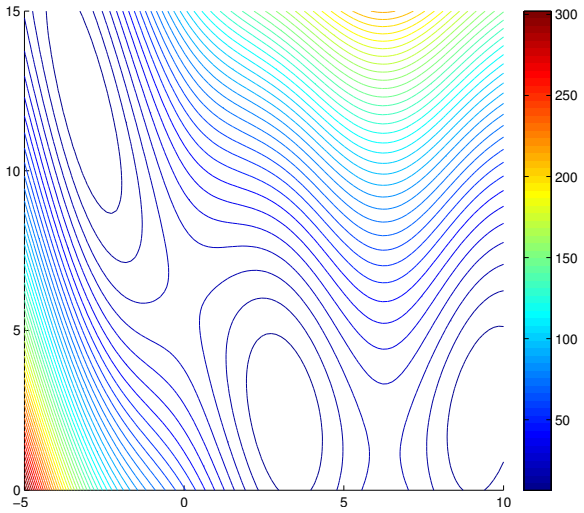

Figure 8.1: *Contours of the Branin function on* $[-5, 10] \times [0, 15]$

the different approximations as well as the different methods for finding the weights from Section 3.3. We use the cubic spline basis function along with a linear polynomial term. The approximation error in different norms is given in Table 8.1 and Figure 8.2 compares the weighted and unweighted surrogates. Observe that the weighted RBF surrogates are much better than the standard RBF surrogate and there appears to be very little difference between the different techniques for finding the weights (visually they are identical). From

Interpolant	w_1^2	w_2^2	$\|s-f\|_1$	$\|s-f\|_2$	$\|s-f\|_\infty$
RBF	1	1	6.757	10.960	106.643
Weighted RBF using MLE (p. 21)	0.849794	$0.3\dot{8}$	3.609	6.889	75.357
Weighted RBF using REML (p. 23)	0.870522	$0.3\dot{8}$	**3.521**	**6.770**	**74.294**
Weighted RBF using LOOCV (p. 25)	0.943073	$0.4\dot{2}5\dot{9}$	3.561	6.823	74.774

Table 8.1: *Norm weights and accuracy of the different approximations in the discretised* L_1, L_2 *and* L_∞ *norms to the Branin function on* $[-5, 10] \times [0, 15]$. *The smallest errors for each norm are denoted in bold.*

an optimisation perspective, this is precisely the sort of behaviour we are looking for in the approximation as we are solely interested in locating the minima. It is therefore clear that any global surrogate optimisation algorithm will fare much better with the weighted RBF interpolant than the standard one as we not only have a better approximation to the function on the entire domain, we have also correctly identified the basins of the minima of the function. For the remainder of the chapter we will use a weighted RBF surrogate with LOOCV as it is the fastest of the methods for finding the weights.

We will now compare the performance of the different branch and bound algorithms proposed in Chapter 6. Let us start by comparing the canonical branch and bound algorithm from Section 6.1 against the proposed Lipschitz branch and bound algorithm from Section 6.5 with overlapping balls and its heuristic counterpart with lattices of balls. We

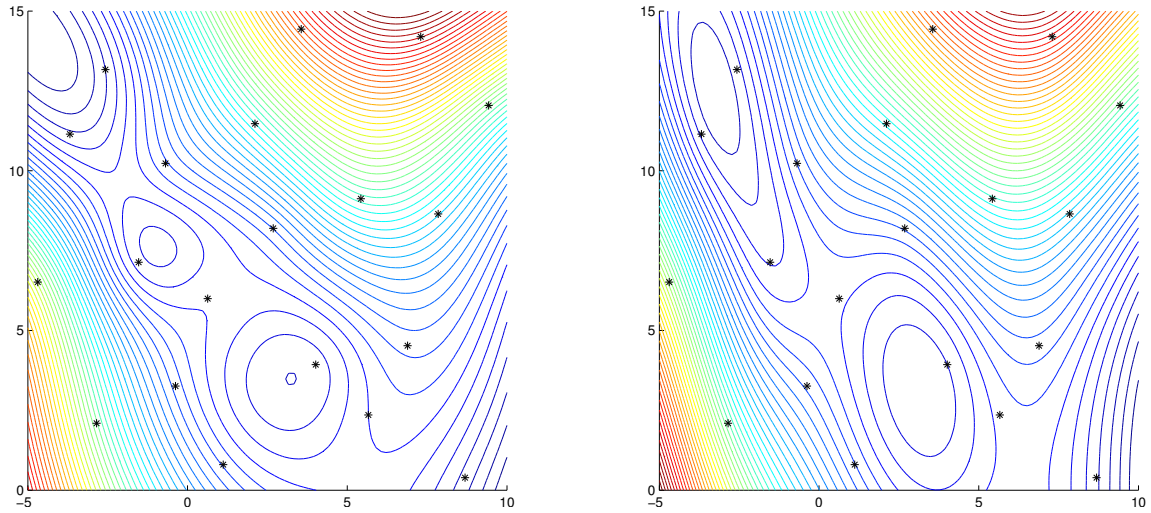

Figure 8.2: *Contours of the unweighted RBF interpolant (left) and weighted RBF interpolant using RMLE (right) to the Branin function on* $[-5, 10] \times [0, 15]$. *Sample points are denoted by black stars.*

will also compare our algorithms to the canonical Lipschitz branch and bound algorithm (see Section 5.3 of Pardalos et al., 1995) which is simply the standard branch and bound algorithm from Chapter 6 with the lower bound

$$\alpha(\mathcal{B}) = f(x_{\mathcal{B}}) - L(\mathcal{B}) \max_{x \in \mathcal{B}} \|x - x_{\mathcal{B}}\| \tag{8.1.1}$$

where $x_{\mathcal{B}}$ is the midpoint of $\mathcal{B}$ and $L(\mathcal{B})$ an upper bound on the optimal surrogate Lipschitz constant over $\mathcal{B}$, calculated by bounding the norm of the gradient over $\mathcal{B}$. For this example, the objective function $f : [-4, 4]^n \subset \mathbb{R}^n \to \mathbb{R}$ will be a sum of sine functions given by

$$f(x) = \sum_{k=1}^{n} \sin x_k$$

and we will optimize a cubic spline surrogate fitted to the objective function at $10n$ maximin Latin hypercube sample points in $[-4, 4]^n$. The surrogate typically has a number of local minima with one global minimum as one can see in Figure 8.3 overleaf for dimension $n = 2$. Table 8.2 overleaf shows the run time in seconds of a Matlab implementation of each branch and bound algorithm for dimensions n from 2 to 5. The algorithm was stopped if it verifiably found the global minimum to within a tolerance of 10^{-2}, i.e. $U_k - L_k < 10^{-2}$. If this was not possible in 3000 seconds (50 minutes) we give the tolerance reached by the algorithm instead. In all cases, the number of iterations of step 1 of the algorithm in question is given in brackets. The experiments were performed on an AMD Phenom II X3 705e processor machine with 4 GB of RAM running Matlab R2010b and the NAG toolbox for Matlab, Mark 22 which provided the local optimization solvers. As one can see from the results, the canonical Lipschitz branch and bound algorithm shows the worst performance and requires considerably more iterations than all the other algorithms. This is because the lower bounding function (8.1.1) used in the algorithm only makes use of the function Lipschitz constant

	2	3	4	5
Canonical	6 (224)	635 (4668)	7×10^{0} (5330)	5×10^{1} (4239)
Canonical Lipschitz	143 (2123)	4×10^{0} (5902)	1×10^{2} (5326)	4×10^{2} (4333)
Lipschitz using Balls	4 (68)	638 (1279)	2×10^{1} (770)	4×10^{3} (182)
Heuristic Lipschitz	**3 (59)**	**68 (342)**	**2461 (2294)**	**4×10^{1} (669)**

Table 8.2: *Run times of the different branch and bound algorithms on a surrogate fitted to the sum of sines function for dimensions from 2 to 5. The tolerance reached is given instead if the algorithm did not complete in time. The number of iterations is given in brackets and the best results for each dimension are denoted in bold.*

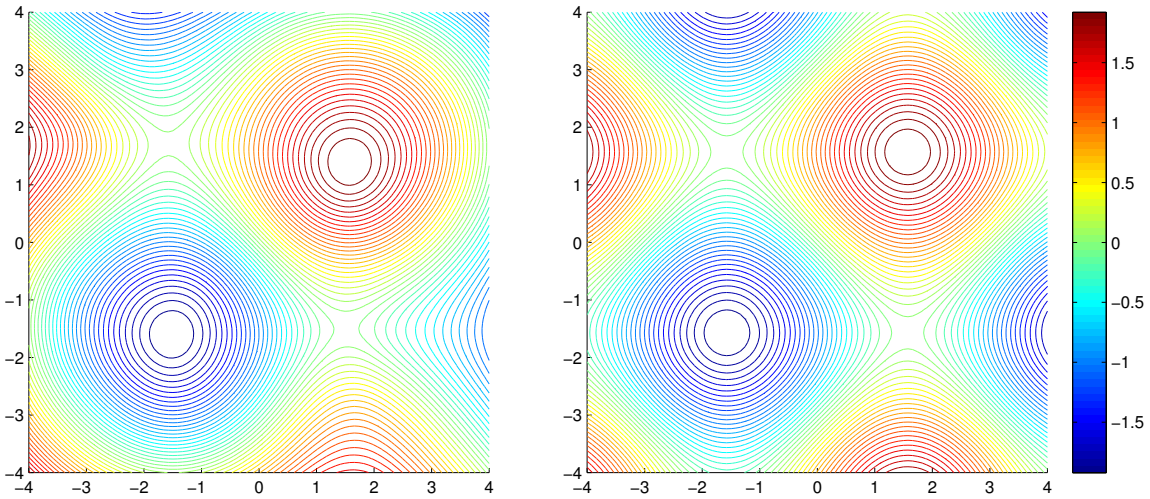

Figure 8.3: *Contours of the surrogate (left) fitted to the sum of sines function (right) on* $[-4,4]\times[-4,4]$. *As one can see the surrogate is particularly accurate for this example.*

and is therefore quite crude. The other algorithms use much tighter bounds which explains their superior performance in lower dimensions. In lower dimensions our proposed Lipschitz algorithms outperform the canonical algorithm as the bounds are tighter and very fast to compute. As the dimension increases, the need to split 3^n balls at each iteration hampers the performance of our Lipschitz algorithm, however the heuristic version (which splits considerably fewer balls at each iteration) consistently outperforms the canonical algorithm. Nonetheless, one must concede that the tolerance returned by the heuristic algorithm is often that of a global minimum on a subset of $[-4,4]^n$, particularly in higher dimensions. While the inability of the algorithms to complete in higher dimensions may seem disappointing one must bear in mind that these are research implementations written in object oriented Matlab which is very slow. For comparison our C++ implementation of the canonical branch and bound algorithm is about 100-200 times faster taking $0.02, 5$ and 152 seconds to find the global minimum for dimensions $n = 2, 3, 4$ in the above example. Moreover, our Lipschitz algorithm can be trivially parallelised as the bounds on each ball can be computed independently which should lead to a significant speedup on modern multiprocessor hardware.

Let us now compare the performance of our proposed branch and bound algorithm for convex constraints from Section 6.2 against our Lipschitz algorithm from Section 6.5 with overlapping balls and its heuristic counterpart which can both handle convex constraints. As in the previous example, the objective function $f : [-4, 4]^n \subset \mathbb{R}^n \to \mathbb{R}$ will be the sum of sine functions, this time with the elliptical constraint

$$c(x) = x^T C x - 1 \leq 0$$

where C is a matrix with $1/2$ on the diagonal and $1/4$ elsewhere. As before, we will optimize a cubic spline surrogate fitted to the objective function at $10n$ maximin Latin hypercube sample points in $[-4, 4]^n$. This time the global minimum of the constrained surrogate typically lies on the boundary as one can see in Figure 8.4. It is our intention to test the convergence of the algorithms on the boundary, for if the global minimum were to lie in the interior, performance would be comparable to the bound constrained example above (see the illustrative numerical examples in Chapter 6). Table 8.3 overleaf shows the run time in

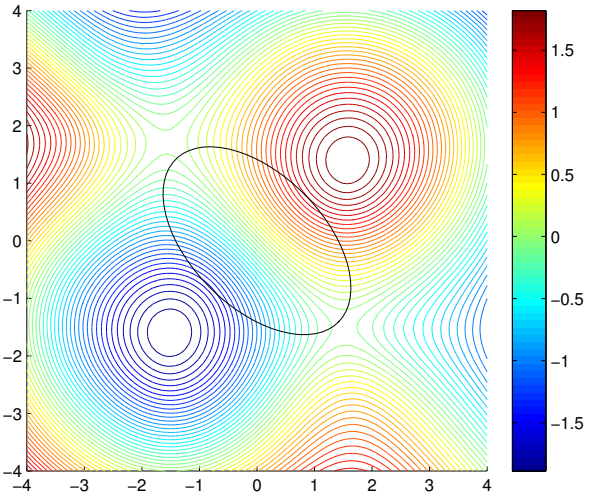

Figure 8.4: *Contours of the surrogate fitted to the sum of sines function on* $[-4, 4] \times [-4, 4]$ *with the elliptical constraint region denoted in black.*

seconds of a Matlab implementation of each constrained branch and bound algorithm for dimensions n from 2 to 5. The testing methodology and hardware is the same as in the previous experiments. One can see that for dimensions greater than 2 our convex constrained algorithm using boxes is faster than the Lipschitz algorithm using balls. One might be tempted to say this is because the convex constrained algorithm is accelerated using constrained local searches but implementing local searches for the Lipschitz algorithms in the same way slows them down considerably. The speed advantage of the heuristic Lipschitz algorithm is evident, once again it consistently outperforms the convex constrained algorithm for dimensions greater than 2. Even so, the results are worse than in the bound constrained example above. This is because the boundary contains a large region where the function takes similar values and the algorithms expend a significant effort searching this region for the global min-

	2	3	4	5
Canonical	4 (105)	1082 (7584)	1×10^{-1} (10412)	4×10^{0} (5648)
Lipschitz using Balls	**4 (56)**	5×10^{-2} (2404)	1×10^{1} (783)	3×10^{1} (299)
Heuristic Lipschitz	4 (94)	**456 (1348)**	$\mathbf{2\times10^{-2}}$**(2372)**	$\mathbf{6\times10^{-2}}$**(1319)**

Table 8.3: *Run times of the different convex constrained branch and bound algorithms on a surrogate fitted to the sum of sines function for dimensions from 2 to 5. The tolerance reached is given instead if the algorithm did not complete in time. The number of iterations is given in brackets and the best results for each dimension are denoted in bold.*

imiser. However, in such situations a constrained local search algorithm can easily locate the global minimum once the branch and bound algorithm has located its basin of attraction.

Finally, we will compare the performance of our one-stage lookahead framework against existing surrogate based global optimization algorithms in the literature. We will use the standard global optimization test functions from Dixon and Szegő (1978) which are widely used in the surrogate based global optimization literature. The test set consists of seven functions with dimensions ranging from two to six all defined on rectangular domains. Table 8.4 below gives a brief overview of the test functions. To facilitate comparisons with existing

	n	Local Minima	Global Minima	$\mathcal{D}$
Branin	2	3	3	$[-5,10]\times[0,15]$
Goldstein-Price	2	4	1	$[-2,2]^2$
Hartman 3	3	4	1	$[0,1]^3$
Shekel 5	4	5	1	$[0,10]^4$
Shekel 7	4	7	1	$[0,10]^4$
Shekel 10	4	10	1	$[0,10]^4$
Hartman 6	6	4	1	$[0,1]^6$

Table 8.4: *The dimension n, number of local and global minima and domain $\mathcal{D}$ for each of the Dixon-Szegő test functions.*

algorithms in the literature, we will stop our one-stage lookahead framework when the relative error $|y_{min} - f^*|/|f^*|$ is less than 1%, where f^* is the global optimum and y_{min} the best function value found so far (note that f^* is non-zero for all the Dixon-Szegő functions). We will compare the cost functions from Section 5.2 using the parameters $\tau = 0.5$ for the scaled lower confidence bound and T chosen according to strategy IIa from Gutmann (2001) for the probability of improvement. In all cases, we start with an initial maximin Latin hypercube design of $10n$ sample points and use the canonical branch and bound algorithm from Section 6.1 as our Lipschitz based algorithm can only handle the surrogate minimum cost function at present. We will also compare against other surrogate based global optimization algorithms from the literature. These are RBF-CUB (Gutmann, 2001) with target selection rule IIIb; rbfSolve (Björkman and Holmström, 2000), an alternative implementation of RBF-CUB; ARBFMIP (Holmström, 2008), a heuristically enhanced version of rbfSolve;

EGO (Jones et al., 1998) and CORS-RBF (Regis and Shoemaker, 2005). It is important to note that all of these surrogate based algorithms (with the exception of CORS-RBF which uses a non-convex constraint region) are special cases of our general framework from Section 5.2, although some of them use different global solvers for optimizing the cost function. For completeness, we also include three global optimization algorithms not based on surrogates. These are DIRECT (Jones et al., 1993), MCS (Huyer and Neumaier, 1999), both branch and bound methods and DE (Storn and Price, 1997), a stochastic search method. Table 8.5 shows the results for our framework, as well as the algorithms from the literature which are taken from the references cited (except the results for DE which are from Huyer and Neumaier, 1999 and MCS from Gutmann, 2001). Note that results for EGO on the Shekel functions were not given in the original paper, presumably because the algorithm ran into difficulties. For the surrogate based algorithms from the literature we give the corresponding cost function in our framework in brackets. The maximum number of iterations for all the surrogate based algorithms is 150, in keeping with the literature.

	BR	G-P	H 3	S 5	S 7	S 10	H 6
Surrogate Minimum (SM)	26	37	32	*	*	*	*7%*
Maximum Error	*	*	*4%*	*	*	*	*
Lower Confidence Bound	28	37	32	*	*	*	*7%*
Probability of Improvement (PI)	36	42	32	*	*	*	78
Expected Improvement (EI)	37	29	32	*	*	*	*7%*
RBF-CUB (PI)	35	55	32	81	65	65	59
rbfSolve (PI)	26	27	22	96	72	76	87
ARBFMIP (Heuristic PI)	22	21	31	34	31	25	43
EGO (EI)	28	32	35				121
CORS-RBF (Constrained SM)	34	49	25	41	46	51	108
DIRECT	63	101	83	103	97	97	213
MCS	31	40	79	83	106	103	74
DE	1190	1018	476	6400	6194	6251	7220

Table 8.5: *Number of samples (including initial samples) required by the algorithms to achieve a relative error of 1%. A star denotes the algorithm was unable to locate the global minimum after 150 iterations. If the algorithm reached this limit but the relative error was less than 10%, it is given in italics. Our cost functions use the canonical branch and bound algorithm from Section 6.1*

Objective functions which have large differences between function values present a problem for surrogate based optimization algorithms (as the surrogate tends to oscillate) and the Goldstein-Price function is one such example (see Gutmann, 2001). To overcome this limitation Gutmann proposes replacing function values greater than the median of all available values by the median and we employ this for the Goldstein-Price function. The other surrogate based algorithms also employ this technique for all the Dixon-Szegő functions with the exception of EGO which uses a log transform for the Goldstein-Price function.

One can see from the results in Table 8.5 that our framework shows similar performance to existing surrogate based algorithms. This should come as no surprise since we are generalising these existing approaches. The differences are mainly due to using alternate parameter values, initial designs, basis functions and in the case of ARBFMIP clever heuristics. The Shekel functions are particularly problematic for some of the methods and this difficulty arises in all surrogate based optimization algorithms. This is because they have many local minimisers which lie at the bottom of very steep, narrow wells (in an otherwise flat surface) and surrogate based algorithms have a tendency to smooth over these wells (see Gutmann, 2001 and Holmström, 2008). Is important to note that the results given in Table 8.5 for RBF-CUB and rbfSolve are the select few of many runs with various parameter values, many of which were unable to locate the global minimum in less than 150 iterations. ARBFMIP on the other hand has a sophisticated routine to deal with this issue which tries various target values and selects only certain points for evaluation. We are confident that given time we could also find a combination of parameters which would locate the global minimum for the Shekel functions, after all most of the surrogate based algorithms from the literature are included in our framework.

8.2 Oil Well Placement

For an industrial application example of our global optimization framework, consider the problem of finding the best place to drill a new oil well in an existing oil reservoir. This is a classic problem in oil reservoir management which is becoming increasingly important as we begin to exhaust existing easily recoverable oil reserves. Typically one has a reservoir for which production from existing oil wells is falling and one seeks to drill a new well to boost production. To tackle this problem in practice, reservoir engineers build numerical models of the underlying oil reservoir and evaluate these for various strategies under consideration to help make an informed decision. However, the optimal strategy may be far from obvious as the model under consideration is relatively complicated, usually a finite volume discretisation which models the flow of oil, water and gas in the porous medium of the reservoir (see Chen, 2007, for an excellent introduction to reservoir simulation). It is natural to formulate the well placement problem as a global optimization problem on a suitable reservoir simulation model. As the simulation model is computationally expensive to evaluate one often approximates its response to the optimization parameters by a surrogate model. We use radial basis functions in our framework but other surrogate models have also been used in the petroleum engineering literature such as neural networks and splines (see Zubarev, 2009 and Yeten, Castellini, Guyaguler and Chen, 2005). Attempts to solve this problem (both with and without the use of surrogate models) have been previously proposed in the petroleum engineering literature by Yeten, Durlofsky and Aziz (2002), Badru and Kabir (2003), Bailey and Couët (2004), Emerick, Silva, Messer, Almeida, Szwarcman, Pacheco and Vellasco (2009) and Bukhamsin, Aziz and Farshi (2010) amongst others. A common problem with these approaches is that often many simulator runs are required to achieve a

good optimum which may only be a local optimum. In particular, Emerick et al. (2009) and Bukhamsin et al. (2010) use genetic algorithms to optimize the reservoir simulation model directly which is very expensive in terms of the number of simulator runs required. Yeten et al. (2002) and Badru and Kabir (2003) attempt to remedy this problem by running the genetic algorithm on a surrogate of the simulation model, however genetic algorithms are not guaranteed to converge to the global optimum. Our one-stage lookahead approach seeks to remedy these shortcomings by attempting to use as few simulator runs as possible while at the same time aiming to find the global optimum. We believe that despite being very ambitious our approach is the best that one can do given the limited amount of information and time (i.e. simulator runs) available.

We will demonstrate the effectiveness of our global optimization framework in this setting by means of a simple numerical example (for more details of our approach see Farmer, Fowkes and Gould, 2010). The aim is to find the optimal location and trajectory of a single multilateral oil producing well on a 3D Cartesian simulation grid, i.e. a single diagonal well with multiple smaller subwells (called laterals) branching off it. We say that a section of the well is completed if it is open to oil flow. For simplicity, we assume the main well (or mainbore) is completed along a single continuous section (typically this will be the section of the well which traverses the actual oil reservoir) and any laterals are completed along their entire length. We parameterise the wells similarly to Yeten et al. (2002) in continuous grid coordinates (i, j, k), i.e. if a point has coordinates $(1.5, 2.5, 3.5)$ in the simulation grid it is in the centre of the $(1, 2, 3)$ grid block. Let $h_0 \in \mathbb{R}^3$ denote the grid coordinates of the mainbore completion heel and $t_0 \in \mathbb{R}^3$ of the mainbore completion toe. Also, let $r_l \in [0, 1]$ denote the relative position of the l-th lateral heel on the mainbore and $t_l \in \mathbb{R}^3$ denote the grid coordinates of the l-th lateral toe. The coordinates of the l-th lateral heel can then be calculated as $h_l = h_0 + r_l(t_0 - h_0)$. This parameterisation is illustrated in Figure 8.5 below. A single multilateral well with L laterals can then be parameterised by

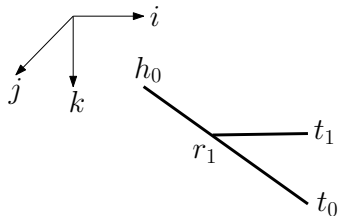

Figure 8.5: *Parameterisation of a multilateral well with a single lateral off the mainbore in continuous grid coordinates i, j, k.*

$x = (h_0, t_0, r_1, t_1, \ldots, r_L, t_L) \in \mathcal{D} \subset \mathbb{R}^{6+4L}$ and these are therefore the variables we wish to optimize over. Let the objective function f for our optimization problem be the negative total oil production from the simulation model over a four year period as a function of the parameters x. The global optimization problem is then to minimize f over $\mathcal{D}$ which is equivalent to maximising the total oil production over a suitably defined rectangular coordinate range $\mathcal{D} \subset \mathbb{R}^{6+4L}$. To this end we will apply our one-stage lookahead framework

from Section 5.2 with the expected improvement cost function and an initial 20 point maximin Latin hypercube design. We will use the synthetic Snark reservoir simulation model, pictured in Figure 8.6 below, which consists of a $24 \times 25 \times 12$ corner point simulation grid representing 12 geological layers with three faults and an analytical aquifer at the southern end of the model. The figure depicts a top down 3D view of the simulation grid with each grid block coloured according to the saturation of oil in that grid block at the beginning of the simulation. The simulation will be performed using the industry standard ECLIPSE

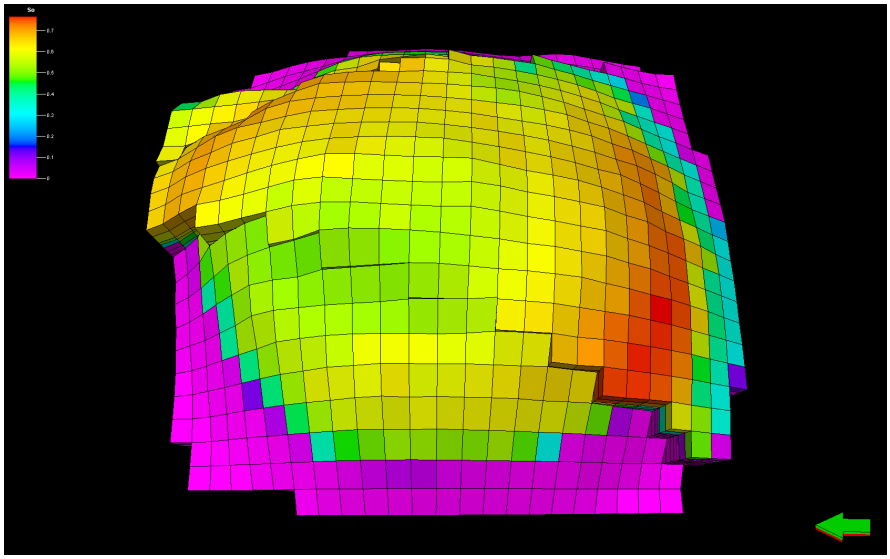

Figure 8.6: *A top-down view of the Snark simulation model showing the initial oil saturation ranging from* 0% *(purple) to* 80% *(red) as indicated by the colourmap in the top left-hand corner.*

simulator from Schlumberger which takes about a minute to run the Snark model over a four year period. Thus each time we try to evaluate the objective function $f(x)$ we call a locally installed version of the simulator which subsequently returns the objective function value. It is important to note that when evaluating the objective function we use our own Matlab interface to the simulator which approximates the corner point grid as a regular Cartesian grid in order to calculate the well connection factors, i.e. how much of the well goes through each simulation grid block. A direct interface to a corner point grid did not exist at the time and would have been too time consuming to develop. This explains the step-wise nature of the optimal well in Figure 8.8.

The aim of the optimization example is to find the optimal mainbore completion for a single multilateral well along with up to three laterals, which means that the optimization problem is in 18 variables. We will compare two different methods for achieving this aim, our surrogate based one-stage lookahead optimization framework and a direct approach using genetic algorithms (cf. Yeten et al., 2002). For our optimization framework we will use the cubic spline radial basis function $\varphi(r) = r^3$ with a diagonally weighted ℓ_2-norm and linear polynomial term, as it is numerically more stable than using the Gaussian radial basis

function (see Section 3.6). We use our own Matlab based implementation of the optimization framework (with a C++ implementation of the canonical branch and bound algorithm) and the genetic algorithm provided by the Matlab global optimization toolbox with default settings. The simulated oil production of the best well found after a given number of simulator runs by our framework and the genetic algorithm is shown in Figure 8.7. For each simulator run $x_i \in \mathcal{D}$, the figure shows the total oil production (over a four year period) of the best producing well found by the algorithm so far, i.e. $\max_{1 \leq k \leq i} f(x_k)$. The total oil production is given in stock tank barrels (STB), which is 42 U.S. gallons of oil stored at 60 °F. One can see that even for this simple example our framework finds a multilateral

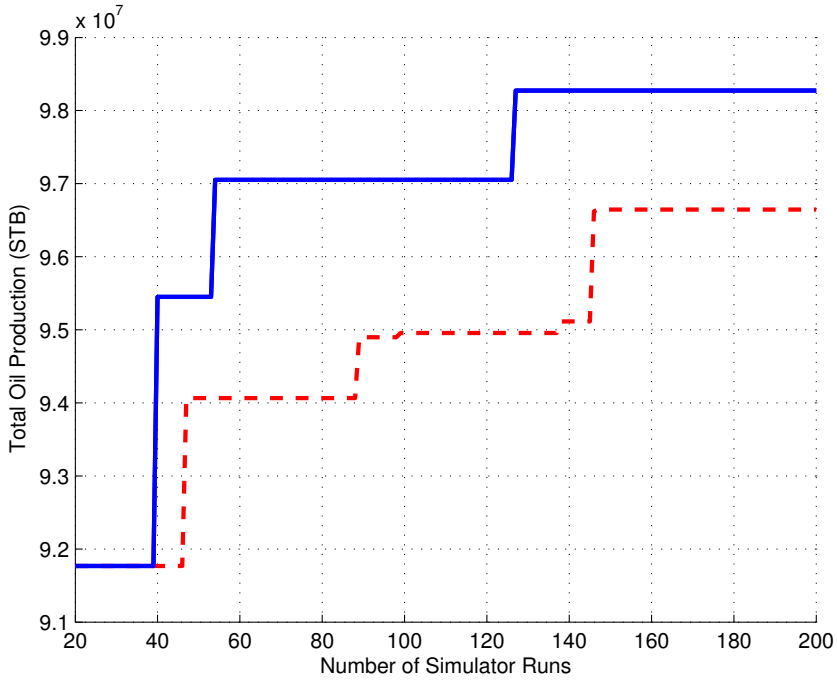

Figure 8.7: *Simulated oil production (in stock tank barrels) of the best multilateral well found after the given number of simulator runs by our framework (solid blue) and the genetic algorithm (dashed red).*

well with consistently better oil production after a given number of simulator runs than the genetic algorithm. While our framework shows only a modest improvement over the genetic algorithm this is financially significant, particularly in today's climate of highly variable oil prices. The trajectory of the optimal well as found by our optimization framework is shown in Figure 8.8 overleaf. The figure depicts a vertical slice through the simulation model given in Figure 8.6 showing the initial oil saturation in each of the grid blocks. Superimposed on the slice in white is the trajectory of the optimal well. This is essentially a horizontal well which maximises contact with areas of high oil saturation and is the type of optimal well one would expect given our choice of simulation model.

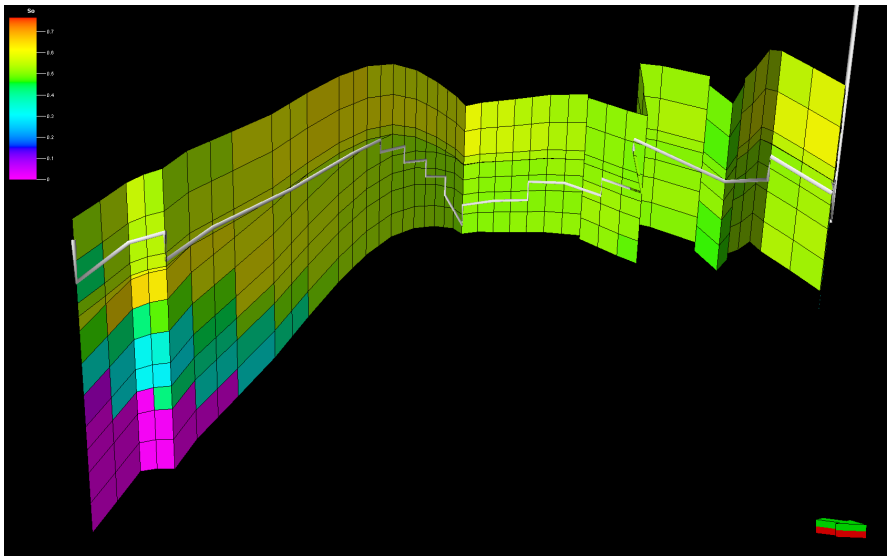

Figure 8.8: *The optimal well trajectory as found by our optimization framework pictured on a vertical slice through the model. The colours depict the initial oil saturation ranging from* 0% *(purple) to* 80% *(red) as indicated by the colourmap in the top left-hand corner.*

Chapter 9

Conclusion

In summary, we have proposed a framework for the global optimization of functions which are expensive to evaluate based on decision theoretic reason. The framework relies on a radial basis function surrogate fitted to scattered data using a weighted ℓ_2-norm along with a measure of the predicted error in the approximation. Using our framework we have generalised standard RBF interpolation (which is obtained as a special case by choosing the weight matrix to be the identity matrix) and shown that all the standard convergence theory pertaining to RBF interpolation holds for this, for lack of a better term, weighted RBF interpolation. Our numerical examples suggest that there are situations which clearly benefit from using weighted RBF surrogates over standard RBF interpolation. Moreover, we have shown that the introduced stochastic interpretation of RBF interpolation enables us to incorporate some of the existing approaches in the literature within our framework as one-stage sampling decisions as well as non-decision theoretic cost functions. This provides a natural, albeit difficult to implement, extension to choosing multiple evaluation points in the form of a multi-stage decision process. It would be of great interest to investigate numerical methods for this extension in the future. In short, our framework gives one a wealth of possible techniques to choose from in addition to the flexibility of using different basis functions such as the numerically more robust low-degree splines. There is also scope for using cost functions other than the ones we have discussed, whether motivated by decision theoretic reasoning or not, provided one is able to optimize them.

Following existing literature, we have adopted a branch and bound approach to globally optimizing the cost functions and proved that the algorithm converges in finite time to within a prescribed tolerance for the cost functions considered. In an attempt to properly handle constraints, we have also extended the algorithm to problems with convex constraints, mixed integer constraints, triangular meshes and the surface of an n-sphere and in all cases proved convergence. In addition we have proposed an entirely new algorithm over convex

sets based on local approximations using Lipschitz constants and proved convergence. The algorithm is currently limited to optimizing the surrogate and we would like to extend it to optimize other cost functions in the future. As the algorithm scales poorly to higher dimensions we have proposed a heuristic version which numerical results suggest is very promising. In all cases, we would like to extend the branch and bound algorithms to handle general nonlinear constraints and investigate the possibility of parallelising them on high-performance computers.

Deciding where to sample the objective function when fitting the surrogate for the first time is an important yet often overlooked aspect of surrogate optimization methods. In order to address this, we have provided a comprehensive review of approaches in the literature to choosing initial designs and compared them numerically to give an indication of their performance. Although the results are inconclusive, the Latin hypercube based designs appear the most promising.

We have shown that our method can also be successfully applied to integrating expensive objective functions. Indeed, in low dimensions this is a promising approach for integration over rectangular domains and spherical surfaces, comparable to Monte Carlo integration.

Finally, as an example industrial application of our method, we implemented our framework on a model problem in oil reservoir simulation and demonstrated that it is a promising approach in this area. We would like to further extend the method in this setting to general unstructured grids which requires a more sophisticated parameterisation of the wells than the simple approach in our example.

In conclusion, there is clearly potential for further improving existing surrogate optimization methodologies both from a theoretical perspective, studying the convergence and behaviour of different approaches and from a practical perspective, designing robust and efficient codes for use on industrial problems.

Bibliography

Ackley, D. H. (1987) *A Connectionist Machine for Genetic Hillclimbing*. Kluwer International Series in Engineering and Computer Science. Kluwer. ISBN 978-0-89838-236-5. http://books.google.co.uk/books?id=3ttfAAAAMAAJ.

Androulakis, I. P., Maranas, C. D. and Floudas, C. A. (1995) 'αBB: A Global Optimization Method for General Constrained Nonconvex Problems'. *Journal of Global Optimization*, vol. 7, pp. 337–363. http://dx.doi.org/10.1007/BF01099647.

Badru, O. and Kabir, C. S. (2003) 'Well Placement Optimization in Field Development'. *SPE 84191 presented at the SPE Annual Technical Conference and Exhibition, Denver, Colorado, 5–8 October*. http://dx.doi.org/10.2118/84191-MS.

Bailey, W. and Couët, B. (2004) 'Field Optimization Tool for Maximizing Asset Value'. *SPE 87026 presented at the SPE Asia Pacific Conference on Integrated Modelling for Asset Management, Kuala Lumpur, Malaysia, 29–30 March*. http://dx.doi.org/10.2118/87026-MS.

Balakrishnan, V., Boyd, S. and Balemi, S. (1991) 'Branch and Bound Algorithm for Computing the Minimum Stability Degree of Parameter-Dependent Linear Systems'. *International Journal of Robust and Nonlinear Control*, vol. 1, no. 4, pp. 295–317. http://dx.doi.org/10.1002/rnc.4590010404.

Bellman, R. E. (2003) *Dynamic Programming*. Dover Books on Mathematics. Dover. ISBN 978-0-486-42809-3. http://books.google.co.uk/books?id=fyVtp3EMxasC.

Berger, J. O. (1985) *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer, second edn. ISBN 978-0-387-96098-2. http://www.springer.com/statistics/statistical+theory+and+methods/book/978-0-387-96098-2.

Bertsekas, D. P. (2005) *Dynamic Programming and Optimal Control*. Athena Scientific Optimization and Computation Series. Athena Scientific. ISBN 978-1-886529-26-7. http://books.google.co.uk/books?id=-E5qQgAACAAJ.

Betrò, B. (1991) 'Bayesian Methods in Global Optimization'. *Journal of Global Optimization*, vol. 1, no. 1, pp. 1–14. http://dx.doi.org/10.1007/BF00120661.

Bishop, C. M. (1996) *Neural Networks for Pattern Recognition*. Oxford University Press. ISBN 978-0-19-853864-6. http://books.google.co.uk/books?id=-aAwQO_-rXwC.

Björkman, M. and Holmström, K. (2000) 'Global Optimization of Costly Nonconvex Functions Using Radial Basis Functions'. *Optimization and Engineering*, vol. 1, no. 4, pp. 373–397. http://dx.doi.org/10.1023/A:1011584207202.

Boyd, S. and Vandenberghe, L. (2004) *Convex Optimization*. Cambridge University Press. ISBN 978-0-521-83378-3. http://www.stanford.edu/~boyd/cvxbook/.

Bratley, P. and Fox, B. L. (1988) 'Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator'. *ACM Transactions on Mathematical Software*, vol. 14, pp. 88–100. http://dx.doi.org/10.1145/42288.214372.

Bratley, P., Fox, B. L. and Niederreiter, H. (1992) 'Implementation and Tests of Low-Discrepancy Sequences'. *ACM Transactions on Modelling and Computer Simulation*, vol. 2, no. 3, pp. 195–213. http://dx.doi.org/10.1145/146382.146385.

Bukhamsin, A. Y., Aziz, K. and Farshi, M. M. (2010) 'Optimization of Multilateral Well Design and Location in a Real Field using a Continuous Genetic Algorithm'. *SPE 136944 presented at the SPE/DGS Saudi Arabia Section Technical Symposium and Exhibition, Al-Khobar, Saudi Arabia, 4–7 April*. http://dx.doi.org/10.2118/136944-MS.

Busby, D., Farmer, C. L. and Iske, A. (2007) 'Hierarchical Nonlinear Approximation for Experimental Design and Statistical Data Fitting'. *SIAM Journal on Scientific Computing*, vol. 29, no. 1, pp. 49–69. http://dx.doi.org/10.1137/050639983.

Caflisch, R. E. (1998) 'Monte Carlo and Quasi-Monte Carlo Methods'. *Acta Numerica*, vol. 7, pp. 1–49. http://dx.doi.org/10.1017/S096249290000280.

Cartis, C., Gould, N. I. M. and Toint, P. L. (2009) 'Adaptive Cubic Regularisation Methods for Unconstrained Optimization. Part I: Motivation, Convergence and Numerical Results'. *Mathematical Programming*. http://dx.doi.org/10.1007/s10107-009-0286-5.

Chen, Z. (2007) *Reservoir Simulation: Mathematical Techniques in Oil Recovery*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM. ISBN 978-0-89871-640-5. http://books.google.co.uk/books?id=eUazfAaGBDEC.

Chilès, J. and Delfiner, P. (1999) *Geostatistics: Modeling Spatial Uncertainty*. Wiley Series in Probability and Statistics. Wiley. ISBN 978-0-471-08315-3. http://books.google.co.uk/books?id=adkSAQAAIAAJ.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (2000) *Trust Region Methods*. MPS-SIAM Series on Optimization. SIAM. ISBN 978-0-89871-460-9. http://books.google.co.uk/books?id=5kNC4fqssYQC.

Conway, J. and Sloane, N. (1999) *Sphere Packings, Lattices, and Groups*, *Grundlehren der mathematischen Wissenschaften*, vol. 290. Springer. ISBN 978-0-387-98585-5. http://www.springer.com/mathematics/algebra/book/978-0-387-98585-5.

Cox, D. D. and John, S. (1997) 'SDO: A Statistical Method for Global Optimization'. In Alexandrov, N. M. and Hussaini, M. (eds.) *Multidisciplinary Design Optimization: State of the Art*, Proceedings in Applied Mathematics Series, No. 80. SIAM. ISBN 978-0-89871-359-6, pp. 315–329. http://books.google.co.uk/books?id=nAh6N5ZXfT4C.

De Marchi, S., Schaback, R. and Wendland, H. (2005) 'Near-Optimal Data-Independent Point Locations for Radial Basis Function Interpolation'. *Advances in Computational Mathematics*, vol. 23, no. 3, pp. 317–330. http://dx.doi.org/10.1007/s10444-004-1829-1.

Di Gaspero, L. (2009) 'QuadProg++'. http://quadprog.sourceforge.net/.

Dixon, L. C. W. and Szegő, G. P. (1978) 'The Global Optimisation Problem: An Introduction'. In Dixon, L. C. W. and Szegő, G. P. (eds.) *Towards Global Optimisation 2*. North-Holland. ISBN 978-0-444-85171-0, pp. 1–15. http://books.google.co.uk/books?id=SUSrAAAAIAAJ.

Emerick, A. A., Silva, E., Messer, B., Almeida, L. F., Szwarcman, D., Pacheco, M. A. C. and Vellasco, M. M. B. R. (2009) 'Well Placement Optimization using a Genetic Algorithm with Nonlinear Constraints'. *SPE 118808 presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, 2–4 Feburary*. http://dx.doi.org/10.2118/118808-MS.

Farmer, C. L., Fowkes, J. M. and Gould, N. I. M. (2010) 'Optimal Well Placement'. *Presented at the 12th European Conference on the Mathematics of Oil Recovery, Oxford, 6 – 9 September*. http://www.earthdoc.org/detail.php?pubid=41313.

Floudas, C., Aggarwal, A. and Ciric, A. (1989) 'Global Optimum Search for Nonconvex NLP and MINLP Problems'. *Computers & Chemical Engineering*, vol. 13, no. 10, pp. 1117–1132. http://dx.doi.org/10.1016/0098-1354(89)87016-4.

Forrester, A., Sóbester, A. and Keane, A. (2008) *Engineering Design via Surrogate Modelling: A Practical Guide*. Progress in Astronautics and Aeronautics. Wiley. ISBN 978-0-470-06068-1. http://books.google.co.uk/books?id=AukeAQAAIAAJ.

Forrester, A. I. J., Sóbester, A. and Keane, A. J. (2007) 'Multi-Fidelity Optimization via Surrogate Modelling'. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, vol. 463, no. 2088, pp. 3251–3269. http://dx.doi.org/10.1098/rspa.2007.1900.

Fox, B. L. (1986) 'Algorithm 647: Implementation and Relative Efficiency of Quasirandom Sequence Generators'. *ACM Transactions on Mathematical Software*, vol. 12, no. 4, pp. 362–376. http://dx.doi.org/10.1145/22721.356187.

Genz, A. (1984) 'Testing Multidimensional Integration Routines'. In Ford, B., Rault, J. C. and Thomasset, F. (eds.) *Proceedings of the International Conference on Tools, Methods and Languages for Scientific and Engineering Computation, May 17-19, Paris, France*. North-Holland. ISBN 978-0-444-87570-9, pp. 81–94. http://books.google.co.uk/books?id=hXFQAAAAMAAJ.

Gill, P. E., Golub, G. H., Murray, W. and Saunders, M. A. (1974) 'Methods for Modifying Matrix Factorizations'. *Mathematics of Computation*, vol. 28, no. 126, pp. 505–535. http://dx.doi.org/10.2307/2005923.

Goldfarb, D. and Idnani, A. (1983) 'A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs'. *Mathematical Programming*, vol. 27, no. 1, pp. 1–33. http://dx.doi.org/10.1007/BF02591962.

Gould, N. I. M., Robinson, D. P. and Thorne, H. S. (2010) 'On Solving Trust-Region and Other Regularised Subproblems in Optimization'. *Mathematical Programming Computation*, vol. 2, no. 1, pp. 21–57. http://dx.doi.org/10.1007/s12532-010-0011-7.

Gramacy, R. B. and Lee, H. K. (2011) 'Optimization Under Unknown Constraints'. In Bernardo, J. M., Bayarri, M. J., Berger, J. O., Dawid, A. P., Heckerman, D., Smith, A. F. M. and West, M. (eds.) *Bayesian Statistics 9: Proceedings of the Ninth Valencia International Meeting, June 3 - 8, 2010*. Oxford University Press (in preparation). http://www.ams.ucsc.edu/share/technical-reports/2010/ucsc-soe-10-14.pdf.

Gutmann, H. M. (2001) *Radial Basis Function Methods for Global Optimization*. Ph.D. thesis, DAMTP, University of Cambridge.

Hager, W. W. and Zhang, H. (2006) 'A New Active Set Algorithm for Box Constrained Optimization'. *SIAM Journal on Optimization*, vol. 17, no. 2, pp. 526–557. http://dx.doi.org/10.1137/050635225.

Halton, J. H. (1960) 'On the Efficiency of Certain Quasi-Random Sequences of Points in Evaluating Multi-Dimensional Integrals'. *Numerische Mathematik*, vol. 2, pp. 84–90. http://dx.doi.org/10.1007/BF01386213.

Hedayat, A., Sloane, N. J. A. and Stufken, J. (1999) *Orthogonal Arrays: Theory and Applications*. Springer Series in Statistics. Springer. ISBN 978-0-387-98766-8. http://www.springer.com/statistics/statistical+theory+and+methods/book/978-0-387-98766-8.

Holmström, K. (2008) 'An Adaptive Radial Basis Algorithm (ARBF) for Expensive Black-Box Global Optimization'. *Journal of Global Optimization*, vol. 41, pp. 447–464. http://dx.doi.org/10.1007/s10898-007-9256-8.

Horst, R. (1986) 'A General Class of Branch-and-Bound Methods in Global Optimization with Some New Approaches for Concave Minimization'. *Journal of Optimization Theory and Applications*, vol. 51, no. 2, pp. 271–291. http://dx.doi.org/10.1007/BF00939825.

Horst, R. and Pardalos, P. M. (1995) *Handbook of Global Optimization*, *Nonconvex Optimization and its Applications*, vol. 2. Springer. ISBN 978-0-7923-3120-9. http://www.springer.com/mathematics/book/978-0-7923-3120-9.

Hubbert, S. and Baxter, B. (2001) 'Radial Basis Functions for the Sphere'. In Haussmann, W., Jetter, K. and Reimer, M. (eds.) *Recent Progress in Multivariate Approximation: 4th International Conference, Witten-Bommerholtz, September 2000*, *International Series of Numerical Mathematics*, vol. 137. Birkhäuser. ISBN 978-3-7643-6505-9, pp. 33–47. http://eprints.bbk.ac.uk/archive/00000396.

Huyer, W. and Neumaier, A. (1999) 'Global Optimization by Multilevel Coordinate Search'. *Journal of Global Optimization*, vol. 14, pp. 331–355. http://dx.doi.org/10.1023/A:1008382309369.

Iske, A. (2000) 'Optimal Distribution of Centers for Radial Basis Function Methods'. Tech. Rep. M0004, Technische Universität München. http://www.math.uni-hamburg.de/home/iske/papers/balance.ps.

Johnson, S. G. (2010) 'The NLopt nonlinear-optimization package'. http://ab-initio.mit.edu/nlopt.

Jones, D. R. (2001) 'A Taxonomy of Global Optimization Methods Based on Response Surfaces'. *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383. http://dx.doi.org/10.1023/A:1012771025575.

Jones, D. R., Perttunen, C. D. and Stuckman, B. E. (1993) 'Lipschitzian Optimization Without the Lipschitz Constant'. *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181. http://dx.doi.org/10.1007/BF00941892.

Jones, D. R., Schonlau, M. and Welch, W. J. (1998) 'Efficient Global Optimization of Expensive Black-Box Functions'. *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492. http://dx.doi.org/10.1023/A:1008306431147.

Kennedy, M. C. and O'Hagan, A. (2000) 'Predicting the Output from a Complex Computer Code when Fast Approximations are Available'. *Biometrika*, vol. 87, no. 1, pp. 1–13. http://dx.doi.org/10.1093/biomet/87.1.1.

Kitanidis, P. K. (1986) 'Parameter Uncertainty in Estimation of Spatial Functions: Bayesian Analysis'. *Water Resources Research*, vol. 22, no. 4, pp. 499–507. http://dx.doi.org/10.1029/WR022i004p00499.

Koehler, J. R. and Owen, A. B. (1996) 'Computer Experiments'. In Ghosh, S. and Rao, C. R. (eds.) *Handbook of Statistics*, vol. 13. Elsevier, pp. 261–308. http://dx.doi.org/10.1016/S0169-7161(96)13011-X.

Kreinovich, V. and Kearfott, R. (2005) 'Beyond Convex? Global Optimization is Feasible Only for Convex Objective Functions: A Theorem'. *Journal of Global Optimization*, vol. 33, pp. 617–624. http://dx.doi.org/10.1007/s10898-004-2120-1.

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*, vol. 86, pp. 97–106.

Lasserre, J. B. (2001) 'Global Optimization with Polynomials and the Problem of Moments'. *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 796–817. http://dx.doi.org/10.1137/S1052623400366802.

Lindley, D. V. (1985) *Making Decisions*. Wiley, second edn. ISBN 978-0-471-90803-6. http://books.google.co.uk/books?id=3-ZQAAAAMAAJ.

Locatelli, M. (1997) 'Bayesian Algorithms for One-Dimensional Global Optimization'. *Journal of Global Optimization*, vol. 10, no. 1, pp. 57–76. http://dx.doi.org/10.1023/A:1008294716304.

Matheron, G. (1962) *Traité de Géostatistique Appliquée, vol. I*, *Mémoires du Bureau de Recherches Géologiques et Miniéres*, vol. 14. Editions Technip, Paris.

Matsumoto, M. and Nishimura, T. (1998) 'Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator'. *ACM Transactions on Modelling and Computer Simulation*, vol. 8, no. 1, pp. 3–30. http://dx.doi.org/10.1145/272991.272995.

McKay, M. D., Conover, W. J. and Beckman, R. J. (1979) 'A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code'. *Technometrics*, vol. 21, no. 2, pp. 239–245. http://dx.doi.org/10.2307/1268522.

Micchelli, C. A. (1986) 'Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions'. *Constructive Approximation*, vol. 2, no. 1, pp. 11–22. http://dx.doi.org/10.1007/BF01893414.

Mockus, J. (1975) 'On Bayesian Methods of Optimisation'. In Dixon, L. C. W. and Szegő, G. P. (eds.) *Towards Global Optimisation*. North-Holland. ISBN 978-0-7204-2463-8, pp. 166–181. http://books.google.co.uk/books?id=U1PvAAAAMAAJ.

Mockus, J. (1989) *Bayesian Approach to Global Optimization: Theory and Applications*, *Mathematics and its Applications*, vol. 37. Kluwer Academic. ISBN 978-0-7923-0115-8. http://books.google.co.uk/books?id=FknvAAAAMAAJ.

Mockus, J. (1994) 'Application of Bayesian Approach to Numerical Methods of Global and Stochastic Optimization'. *Journal of Global Optimization*, vol. 4, no. 4, pp. 347–365. http://dx.doi.org/10.1007/BF01099263.

Mockus, J., Tiesis, V. and Žilinskas, A. (1978) 'The Application of Bayesian Methods for Seeking the Extremum'. In Dixon, L. C. W. and Szegő, G. P. (eds.) *Towards Global Optimisation 2*. North-Holland. ISBN 978-0-444-85171-0, pp. 117–128. http://books.google.co.uk/books?id=SUSrAAAAIAAJ.

Morris, M. D., Mitchell, T. J. and Ylvisaker, D. (1993) 'Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction'. *Technometrics*, vol. 35, no. 3, pp. 243–255. http://dx.doi.org/10.2307/1269517.

Munkres, J. (1997) *Topology: A First Course*. Prentice Hall, second edn. ISBN 978-981-3076-82-2. http://books.google.co.uk/books?id=zLY6PQAACAAJ.

Nesterov, Y. and Polyak, B. (2006) 'Cubic Regularization of Newton Method and its Global Performance'. *Mathematical Programming*, vol. 108, no. 1, pp. 177–205. http://dx.doi.org/10.1007/s10107-006-0706-8.

Neumaier, A. (2004) 'Complete Search in Continuous Global Optimization and Constraint Satisfaction'. *Acta Numerica*, vol. 13, pp. 271–369. http://dx.doi.org/10.1017/S0962492904000194.

Niederreiter, H. (1992) *Random Number Generation and Quasi-Monte Carlo Methods*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM. ISBN 978-0-89871-295-7. http://books.google.co.uk/books?id=pca2tPZozXcC.

O'Hagan, A. (1978) 'Curve Fitting and Optimal Design for Prediction'. *Journal of the Royal Statistical Society B*, vol. 40, no. 1, pp. 1–42. http://dx.doi.org/10.2307/2984861.

O'Hagan, A. (1991) 'Bayes-Hermite Quadrature'. *Journal of Statistical Planning and Inference*, vol. 29, no. 3, pp. 245–260. http://dx.doi.org/10.1016/0378-3758(91)90002-V.

O'Hagan, A. (1992) 'Some Bayesian Numerical Analysis'. In Bernardo, J. M., Berger, J. O. and DeGroot, M. H. (eds.) *Bayesian Statistics 4: Proceedings of the Fourth Valencia International Meeting, April 15-20, 1991*. Oxford University Press. ISBN 978-0-19-852266-9, pp. 345–363. http://books.google.co.uk/books?id=5x3vAAAAMAAJ.

Osborne, M. A., Garnett, R. and Roberts, S. J. (2009) 'Gaussian Processes for Global Optimization'. *Presented at the 3rd International Conference on Learning and Intelligent Optimization (LION3), Trento, Italy, 14 – 18 January*. http://www.robots.ox.ac.uk/~parg/pubs/OsborneGarnettRobertsGPGO.pdf.

Owen, A. B. (1992) 'Orthogonal Arrays for Computer Experiments, Integration and Visualization'. *Statistica Sinica*, vol. 2, no. 2, pp. 439–452. http://www3.stat.sinica.edu.tw/statistica/oldpdf/A2n27.pdf.

Papoulis, P. (1991) *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Series in Electrical Engineering. McGraw-Hill, third edn. ISBN 978-0-07-048477-1. http://books.google.co.uk/books?id=4IwQAQAAIAAJ.

Pardalos, P. M., Horst, R. and Thoai, N. V. (1995) *Introduction To Global Optimization*, *Nonconvex Optimization and its Applications*, vol. 3. Springer. ISBN 978-0-7923-3556-6. http://www.springer.com/mathematics/book/978-0-7923-3556-6.

Pardalos, P. M. and Romeijn, H. E. (2002) *Handbook of Global Optimization Volume 2*, *Nonconvex Optimization and its Applications*, vol. 62. Springer. ISBN 978-1-4020-0632-6. http://www.springer.com/mathematics/book/978-1-4020-0632-6.

Pedamallu, C. S., Özdamar, L., Csendes, T. and Vinkó, T. (2008) 'Efficient Interval Partitioning for Constrained Global Optimization'. *Journal of Global Optimization*, vol. 42, no. 3, pp. 369–384. http://dx.doi.org/10.1007/s10898-008-9297-7.

Pinter, J. D. (1996) *Global Optimization in Action*, *Nonconvex Optimization and its Applications*, vol. 6. Springer. ISBN 978-0-7923-3757-7. http://www.springer.com/mathematics/book/978-0-7923-3757-7.

Rasmussen, C. E. and Williams, C. K. I. (2006) *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press. ISBN 978-0-262-18253-9. http://www.gaussianprocess.org/gpml/chapters/RW.pdf.

Regis, R. G. and Shoemaker, C. A. (2005) 'Constrained Global Optimization of Expensive Black Box Functions using Radial Basis Functions'. *Journal of Global Optimization*, vol. 31, pp. 153–171. http://dx.doi.org/10.1007/s10898-004-0570-0.

Riley, J. D. (1955) 'Solving Systems of Linear Equations With a Positive Definite, Symmetric, but Possibly Ill-Conditioned Matrix'. *Mathematical Tables and Other Aids to Computation*, vol. 9, no. 51, pp. 96–101. http://dx.doi.org/10.2307/2002065.

Rippa, S. (1999) 'An Algorithm for Selecting a Good Value for the Parameter c in Radial Basis Function Interpolation'. *Advances in Computational Mathematics*, vol. 11, no. 2, pp. 193–210. http://dx.doi.org/10.1023/A:1018975909870.

Santner, T. J., Williams, B. J. and Notz, W. (2003) *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. Springer. ISBN 978-0-387-95420-2. http://www.springer.com/statistics/statistical+theory+and+methods/book/978-0-387-95420-2.

Sasena, M. J. (2002) *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. Ph.D. thesis, University of Michigan. http://www.mat.univie.ac.at/~neum/glopt/mss/Sas02.pdf.

Schaback, R. (1993) 'Comparison of Radial Basis Function Interpolants'. In Jetter, K. and Utreras, F. (eds.) *Multivariate Approximation: From CAGD to Wavelets : Proceedings of*

the International Workshop : Santiago, Chile, 24–30 September 1992, *Series in Approximations and Decompositions*, vol. 3. World Scientific. ISBN 978-981-02-1442-5, pp. 293–305. http://books.google.co.uk/books?id=_8F0QgAACAAJ.

Schonlau, M. (1997) *Computer Experiments and Global Optimization*. Ph.D. thesis, University of Waterloo. http://hdl.handle.net/10012/190.

Spall, J. C. (2003) *Introduction to Stochastic Search and Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley. ISBN 978-0-471-33052-3. http://books.google.co.uk/books?id=f66OIvvkKnAC.

Stephens, C. P. and Baritompa, W. (1998) 'Global Optimization Requires Global Information'. *Journal of Optimization Theory and Applications*, vol. 96, no. 3, pp. 575–588. http://dx.doi.org/10.1023/A:1022612511618.

Storn, R. and Price, K. (1997) 'Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces'. *Journal of Global Optimization*, vol. 11, pp. 341–359. http://dx.doi.org/10.1023/A:1008202821328.

Streltsov, S. and Vakili, P. (1999) 'A Non-myopic Utility Function for Statistical Global Optimization Algorithms'. *Journal of Global Optimization*, vol. 14, pp. 283–298. http://dx.doi.org/10.1023/A:1008284229931.

Tang, B. (1993) 'Orthogonal Array-Based Latin Hypercubes'. *Journal of the American Statistical Association*, vol. 88, no. 424, pp. 1392–1397. http://dx.doi.org/10.2307/2291282.

Törn, A. and Žilinskas, A. (1989) *Global Optimization*, *Lecture Notes in Computer Science*, vol. 350. Springer. ISBN 978-3-540-50871-7. http://dx.doi.org/10.1007/3-540-50871-6.

Turk, G. and Levoy, M. (1994) 'Zippered Polygon Meshes from Range Images'. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94. ACM. ISBN 978-0-89791-667-7, pp. 311–318. http://dx.doi.org/10.1145/192161.192241.

Vavasis, S. (1991) *Nonlinear Optimization: Complexity Issues*, *International Series of Monographs on Computer Science*, vol. 8. Oxford University Press. ISBN 978-0-19-507208-2. http://books.google.co.uk/books?id=AdlQAAAAMAAJ.

Vazquez, E. and Bect, J. (2010) 'Convergence Properties of the Expected Improvement Algorithm with Fixed Mean and Covariance Functions'. *Journal of Statistical Planning and Inference*, vol. 140, no. 11, pp. 3088 – 3095. http://dx.doi.org/10.1016/j.jspi.2010.04.018.

Ver Hoef, J. and Cressie, N. (1993) 'Multivariable Spatial Prediction'. *Mathematical Geology*, vol. 25, pp. 219–240. http://dx.doi.org/10.1007/BF00893273.

Žilinskas, A. (2010) 'On Similarities Between Two Models of Global Optimization: Statistical Models and Radial Basis Functions'. *Journal of Global Optimization*, vol. 48, pp. 173–182. http://dx.doi.org/10.1007/s10898-009-9517-9.

Weiss, N. A. (2005) *A Course in Probability*. Addison Wesley. ISBN 978-0-321-18954-7. http://books.google.co.uk/books?id=Be9fJwAACAAJ.

Wendland, H. (2005) *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press. ISBN 978-0-521-84335-5. http://books.google.co.uk/books?id=qy4cbWUmSyYC.

Wikramaratna, R. S. (1989) 'ACORN – A New Method for Generating Sequences of Uniformly Distributed Pseudo-Random Numbers'. *Journal of Computational Physics*, vol. 83, no. 1, pp. 16–31. http://dx.doi.org/10.1016/0021-9991(89)90221-0.

Wikramaratna, R. S. (2008) 'The Additive Congruential Random Number Generator – A Special Case of a Multiple Recursive Generator'. *Journal of Computational and Applied Mathematics*, vol. 216, no. 2, pp. 371–387. http://dx.doi.org/10.1016/j.cam.2007.05.018.

Wolsey, L. A. (1998) *Integer Programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley. ISBN 978-0-471-28366-9. http://books.google.co.uk/books?id=x7RvQgAACAAJ.

Yeten, B., Castellini, A., Guyaguler, B. and Chen, W. (2005) 'A Comparison Study on Experimental Design and Response Surface Methodologies'. *SPE 93347 presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, 31 January – 2 Feburary*. http://dx.doi.org/10.2118/93347-MS.

Yeten, B., Durlofsky, L. J. and Aziz, K. (2002) 'Optimization of Nonconventional Well Type, Location and Trajectory'. *SPE 77565 presented at the SPE Annual Technical Conference and Exhibition, San Antonio, Texas, 29 September – 2 October*. http://dx.doi.org/10.2118/77565-MS.

Zubarev, D. I. (2009) 'Pros and Cons of Applying Proxy-models as a Substitute for Full Reservoir Simulations'. *SPE 124815 presented at the SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, 4–7 October*. http://dx.doi.org/10.2118/124815-MS.