Bayesian Optimisation for Vector-Valued Functions

Jenneskens, T.C.W.

Award date:
2024

Bayesian Optimisation for Vector-Valued Functions

T.C.W. Jenneskens (0957331)
Department of Mathematics and Computer Science
Eindhoven University of Technology

Supervisors:
B. van der Linden
L. Swaenen
G. Bollen
B. Koren
F. Röttger
O. Mula Hernández

March 7, 2024

Bayesian Optimisation is a method for the global optimisation of univariate black-box functions, in which sequentially a Gaussian Process is constructed to model the unknown function, and an acquisition function is optimised in order to pick the next sample point. The aims of this thesis are twofold: firstly, we investigate the extension of Bayesian Optimisation to multivariate functions, which are to be minimised with respect to the maximum absolute-value norm. To facilitate this, we derive acquisition functions that incorporate information from all function components at the same time, and we numerically test this against regular Bayesian Optimisation. Secondly, we address the challenge of globally optimising acquisition functions, an oft-neglected aspect of Bayesian Optimisation, by suggesting modified acquisition functions that are Lipschitz-continuous. Using this property, we propose an algorithm that enables us to get arbitrarily close to their global optima.

Contents

1. Introduction

Suppose we want to minimise a black-box function f, that is, a function we know nothing about, on a compact set. All we are able to do is sample its values, but such a sampling may be costly. How can we find the minimal cost using as few samples as possible? A possible approach is given by Bayesian Optimisation.
The aim of this thesis is to investigate Bayesian Optimisation methods to solve the composite function minimisation problem

$$\min_{x \in \Omega} g(f(x)),$$

where $f : \mathbb{R}^m \to \mathbb{R}^n$ is a black-box function that is expensive to evaluate, $g : \mathbb{R}^n \to \mathbb{R}$ is also black-box, but cheaper to evaluate, and $\Omega \subset \mathbb{R}^m$ is the **bounded** search space. These functions are 'black-box functions' in the sense that we know nothing of their inner working. All we are able to do is compute the output at any point in the input space, but each of these computations is costly. The core idea of Bayesian Optimisation is to evaluate $g \circ f$ at a few points, and use this information to model the output on the whole search space as a Gaussian Process by using Bayes' Rule [Frazier, 2018]. This leads to a model

$$g(f(x)) \sim \mathcal{N}(\mu(x), \sigma^2(x)) \text{ for all } x \in \Omega,$$

where $\mathcal{N}(\mu, \sigma^2)$ denotes a normal distribution with mean μ and standard deviation σ. The mean function $\mu(\cdot)$ is an estimate of the value of the composite function at every point, and the function $\sigma(\cdot)$ quantifies the uncertainty in that estimate. In this way, Bayesian Optimisation allows the balancing of exploitation (looking for points with low expected value) against exploration (looking for points with high uncertainty). Choosing the next point to evaluate is done by optimising a heuristic called an acquisition function, which is defined based on the modelled distribution. Some common choices for this heuristic are the lower confidence bound

$$\mathrm{LCB}(x) := \mu(x) - \sigma(x),$$

the probability of improvement

$$\mathrm{PI}(x) := \mathbb{P}\left(g(f(x)) \leq y^*\right)$$

and the expected improvement

$$\mathrm{EI}(x) := \mathbb{E}\left[\max\left(y^* - g(f(x)), 0\right)\right].$$

Here y^* is the minimal composite function value found so far. These acquisition functions will be introduced in greater detail in 2.3.
However, we are missing out on free information in the abovementioned approach. When evaluating the composite function $g \circ f$, we might as well keep track of the outputs of the inner function f. Some prior research has been done on methods that extend Bayesian Optimisation to the composite function case.

In [Uhrenholt and Jensen, 2019], f is a vector-valued function and g is the Euclidean distance to a target vector y^*. First each component of f is modelled as an independent Gaussian Process, then the authors approximate the distribution of the sum of squares of these Gaussians by a non-central χ^2-distribution, and finally this distribution is approximated by a normal distribution. The authors then go on to derive closed-form expressions for the expected improvement and lower confidence bound acquisition functions and their derivatives, which allows for easy optimisation. Finally, they compare their methods empirically to standard methods and observe improvement both on synthetic benchmark problems and on a practical problem.

In [Astudillo and Frazier, 2019], f is a black-box expensive vector-valued function and g is a cheap real-valued function. The authors model f as a multi-output Gaussian Process (so the components can be correlated) and use a Monte Carlo approach to approximate the expected improvement in the composite function, as well as the gradient of this acquisition function. They then perform numerical experiments that indicate the approach can outperform standard Bayesian Optimisation.

In this thesis, we focus on the case where $f : \mathbb{R}^m \to \mathbb{R}^n$ is a black-box expensive vector-valued function and g is the maximum absolute value norm

$$g(x) := \|f(x)\|_\infty = \max_{1 \leq i \leq n} \left| f_i(x) \right|.$$

As in [Uhrenholt and Jensen, 2019] we model each component f_i as an independent Gaussian Process on the input space Ω. Probability theory is used to derive a closed-form expression for the probability of improvement.[1] We use the Julia package GaussianProcesses.jl to model the components as Gaussian Processes and compare our approach against standard Bayesian Optimisation (only using the output of the composite function as a whole) and Nelder-Mead on two benchmark problems and the practical application BendAid.

The main research question this report sets out to answer is "How does the method described above compare to existing methods on the problem of minimisation with respect to the maximum absolute value norm?". We will make this comparison on a benchmark problem and on a real-world problem. The benchmark problem is to minimise a function which has arbitrary input and output dimension. In this way, the methods will be compared on these problems for two different dimensional setups. The real-world problem has large output dimension when all the data is used, but this can also be scaled down. The comparisons will be made in terms of the number of necessary function evaluations and in terms of the processing time required. These criteria tell a different story; the number of function evaluations is more interesting in the case where the function becomes very expensive to evaluate, meaning it costs a lot of computing time and possibly memory. In contrast, the total processing time is

[1] Not strictly closed-form, this does rely on the standard normal cumulative distribution function.

more interesting when the function is not that expensive to evaluate, as it incorporates the time it takes to make a model for each function component.

2. Preliminaries

In this section, we introduce the framework of Bayesian Optimisation. The problem we want to solve by applying Bayesian Optimisation is

$$\min_{x \in \Omega} f(x),$$

where $f : \Omega \to \mathbb{R}$ is a function defined on a compact subset $\Omega \subset \mathbb{R}^d$. We consider f as a black-box function, meaning we have no knowledge about any structure it might have. Furthermore, we assume evaluating f at points $x \in \Omega$ is costly.
We start in Subsection 2.1 by introducing Gaussian Process Regression, the method we use to model f as an infinite-dimensional distribution. Then we dive into the details of different aspects in the rest of this section. In Subsection 2.2, we derive the conditional distribution to predict unknown data based on known data. Using this prediction, we want to choose the next point to sample. Subsection 2.3 discusses how acquisition functions enable us to do this, and gives a few examples of commonly used acquisition functions. We finish with Subsection 2.4, in which we examine the kernel function. The kernel function forms the basis of the covariance the posterior distribution will have, and we consider the properties that are required to ensure this covariance matrix will be invertible.

2.1. Gaussian Process Regression

Gaussian Process Regression is a way to model an unknown function f as a Gaussian Process, incorporating known function values.

Definition 2.1 (Gaussian Process)**.** Let $\Omega \subset \mathbb{R}^d$ be an index set. A Gaussian Process is an infinite collection of random variables $\{X_t : t \in \Omega\}$ such that any finite subset $\{X_{t_1}, \dots, X_{t_n}) \ (t_1, \dots, t_n \in \Omega\}$ is a multivariate Gaussian random vector.

A Gaussian random vector is characterised by its mean vector μ and covariance matrix Σ. In the case of a Gaussian Process, these can be extended to functions of the index set. That way, for a finite subset of the Gaussian Process, the mean vector and covariance matrix can be calculated directly by evaluating these functions on the index set.

Example 2.1 (Gaussian Process)**.** We could choose as our index set $\Omega = [0, 5] \subset \mathbb{R}$, and as our mean and covariance functions

$$\begin{aligned} \mu : \Omega \mapsto \mathbb{R} \text{ given by } \mu = 0, \text{ and} \\ \Sigma : \Omega \times \Omega \mapsto \mathbb{R} \text{ given by } \Sigma(t_1, t_2) = \exp\Big(- \frac{|t_1 - t_2|^2}{2} \Big). \end{aligned}$$

Then if we **evaluate** at $\{t_1, t_2, t_3\} = \{0, 2, 5\} \subset \Omega$, the resulting Gaussian vector X_{t_1,t_2,t_3} will have mean vector $\mathbf{0}$ and covariance matrix

$$\begin{pmatrix} 1 & \exp\Big(-2\Big) & \exp\Big(-\frac{25}{2}\Big) \\ \exp\Big(-2\Big) & 1 & \exp\Big(-\frac{9}{2}\Big) \\ \exp\Big(-\frac{25}{2}\Big) & \exp\Big(-\frac{9}{2}\Big) & 1 \end{pmatrix}.$$

For Gaussian Process Regression, we consider an optimisation problem with d input dimensions and one output dimension. Here, we want to predict the values of a function $f : \mathbb{R}^d \to \mathbb{R}$, where we are given the data

$$\mathcal{D} := \{(x_i, f(x_i)) | i = 1 \dots n\},$$

i.e. n observations of the function f in locations $x_1, \dots, x_n$. The approach is to model f as a Gaussian Process, i.e. for any finite number of points $\{x_1, \dots, x_n\} \subset \mathbb{R}^d$, $(f(x_1), \dots, f(x_n))$ is a Gaussian random vector. At the start of the regression, the Gaussian Process has a mean defined by a function $\mu : \mathbb{R}^d \to \mathbb{R}$ and a covariance defined by a function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. We adopt compact notation similar to the notation used in [Frazier, 2018]: for $1 \leq k \leq n + 1$, $x_{1:k}$ denotes the sequence $x_1, \dots, x_k$ and $f_{1:k}$ denotes the column vector $[f(x_1) \cdots f(x_k)]^\top$. Likewise, $\mu_{1:k} := [\mu(x_1) \cdots \mu(x_k)]^\top$ and $K_{1:k}$ denotes the $k \times k$ matrix

$$\begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_k) \\ \vdots & \ddots & \vdots \\ K(x_k, x_1) & \cdots & K(x_k, x_k) \end{bmatrix}.$$

The priors encode our beliefs about the function values prior to taking $\mathcal{D}$ into account. Therefore, before incorporating any data, we assume the joint function values to be distributed as

$$f_{1:k} \sim \mathcal{N}\Big(\mu_{1:k}, K_{1:k}\Big).$$

To incorporate $\mathcal{D}$, we are interested in the conditional distribution of $f_{n+1}|f_{1:n}$, that is, the distribution of f_{n+1} when we already know the values of $f_{1:n}$. By the law of conditional probability, the density of this conditional distribution is given by

$$p(f_{n+1}|f_{1:n}) = \frac{p(f_{1:n+1})}{p(f_{1:n})}.$$

Writing out these density functions, this becomes

$$p(f_{n+1}|f_{1:n}) = \frac{\sqrt{(2\pi)^n |K_{1:n}|}}{\sqrt{(2\pi)^{n+1} |K_{1:n+1}|}} \cdot \frac{\exp\Big(-(f_{1:n+1} - \mu_{1:n+1})^\top K_{1:n+1}^{-1} (f_{1:n+1} - \mu_{1:n+1})/2\Big)}{\exp\Big(-(f_{1:n} - \mu_{1:n})^\top K_{1:n}^{-1} (f_{1:n} - \mu_{1:n})/2\Big)}. \quad (1)$$

Simplifying this expression involves calculating the inverse of the 2×2 block matrix

$$K_{1:n+1} = \begin{bmatrix} K_{1:n} & K(x_{1:n}, x_{n+1}) \\ K(x_{n+1}, x_{1:n}) & K(x_{n+1}, x_{n+1}) \end{bmatrix},$$

where $K(x_{n+1}, x_{1:n})$ is shorthand for the row vector $\begin{bmatrix} K(x_{n+1}, x_1) & \cdots & K(x_{n+1}, x_n) \end{bmatrix}$ and $K(x_{1:n}, x_{n+1})$ is its transpose, in terms of its blocks. But how do we know this matrix is invertible? For now we will just assume it is, and in the next subsection we will see that we can guarantee this by making an appropriate choice for the kernel function. If the matrix $K_{1:n+1}$ is invertible, then its inverse is known in terms of the inverse of $K_{1:n}$ and the other blocks. [Lu and Shiou, 2002] show that the inverse of a 2×2 block matrix

$$R = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

is given by

$$R^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B(R/A)^{-1}CA^{-1} & -A^{-1}B(R/A)^{-1} \\ -(R/A)^{-1}CA^{-1} & (R/A)^{-1} \end{bmatrix},$$

provided that both R and A are invertible. Here R/A denotes the Schur Complement $D - CA^{-1}B$ of A.
Using this, we are able to simplify (1). For a full derivation, see [Soch, 2020]. We find that the conditional distribution is itself a normal distribution

$$f_{n+1}|f_{1:n} \sim \mathcal{N}\Big(\mu_*, K_*\Big),$$

with mean μ_* and variance K_* given by

$$\mu_* = \mu_{n+1} + K(x_{n+1}, x_{1:n})K_{1:n}^{-1}\Big(f_{1:n} - \mu_{1:n}\Big), \tag{2}$$

$$K_* = K(x_{n+1}, x_{n+1}) - K(x_{n+1}, x_{1:n})K_{1:n}^{-1}K(x_{1:n}, x_{n+1}). \tag{3}$$

Example 2.2 (Gaussian Process Regression)**.** In Figure 2.2, we see an illustration of Gaussian Process Regression. For this example, the prior functions chosen were

$$\mu_0(x) := 0, \text{ and}$$
$$K_0(x, y) := \exp\Big(-\frac{|x - y|^2}{2}\Big).$$

The black curve represents the target function $f(x) = e^{-(x-1)^2/2} + 2e^{-(x-4)^2}$ we want to minimise. The function has been evaluated at the five points $(0, \frac{5}{4}, \frac{5}{2}, \frac{15}{4}, 5)$ shown as red dots in the graph. The red line and blue surface represent the conditional distribution of f based on these five known data points: The red curve is the posterior mean $\mu_*(x)$ and the blue area is the '95 % certainty interval' enclosed by the curves $\mu_*(x) + 2\sigma_*(x)$ and $\mu(x) - 2\sigma_*(x)$, where $\sigma_*(x) := \sqrt{K_*(x, x)}$ is the posterior standard deviation.

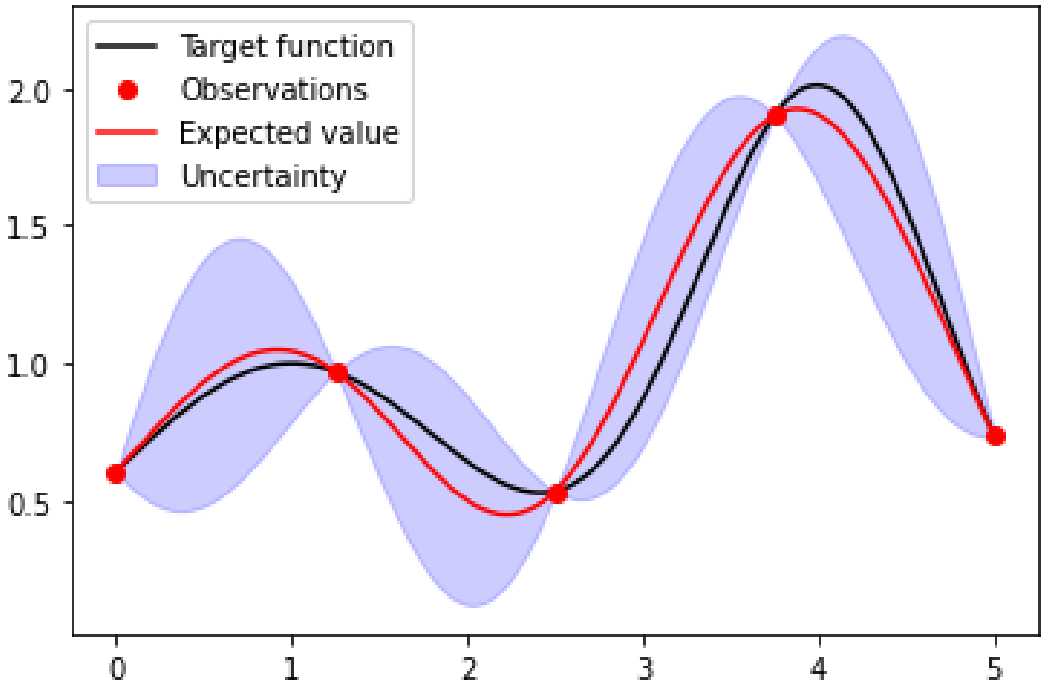

Figure 2.1: An illustration of Gaussian Process Regression.

In Subsection 2.4, we will return to the claim that we can choose a kernel function which always generates invertible matrices to see how to achieve this. We will also look at kernel choices which additionally allow us to generate sparse matrices, potentially speeding up the computations. But first, we will turn our attention to acquisition functions in the next subsection, to finish this introduction to Bayesian Optimisation.

2.2. Bayesian Optimisation

Suppose we want to minimise a black-box function $f : \Omega \to \mathbb{R}$, that is, a function we know nothing about, on a compact set $\Omega \subset \mathbb{R}^d$. All we are able to do is sample its values, but such a sampling is costly: for example, one could think of designing a bridge, where the function f denotes the total economic cost of building the bridge. How can we find the minimal cost in as few bridges built as possible? A possible approach is given by Bayesian Optimisation. The idea of Bayesian Optimisation, in a nutshell, is as follows:

1. we sample f at some starting points $(x_1, \ldots, x_n)$. A common choice for a d-dimensional input space Ω is to start with $d + 1$ points. A method like Latin Hypercube Sampling [Bates et al., 2004] can be employed to ensure a good starting representation of the input space.

2. We apply Gaussian Process Regression (see Subsection 2.1) to create a surrogate model of f, incorporating the knowledge we have of f so far.

3. We select $x_{n+1} \in \Omega$ based on a selection criterion and sample f at this point.

4. We repeat steps 2 and 3 until we encounter a stopping criterion.

Now let's go into these steps in a bit more detail. Step 1 is self-explanatory.

In step 2, we model f as a Gaussian Process. We start by making prior assumptions about the mean and covariance of this distribution. To do this, we choose two functions: a mean function $\mu : \Omega \to \mathbb{R}$ and a kernel function $K : \Omega \times \Omega \to \mathbb{R}$. Here the kernel function will determine the covariance of the Gaussian Process. For points $x_1, \ldots, x_n \in \Omega$, we assume that, a priori, these functions describe the distribution of the random vector $(f(x_1), \ldots, f(x_n))$ at those points. $(f(x_1), \ldots, f(x_n))$ will follow a normal distribution with mean vector

$$(\mu(x_1), \ldots, \mu(x_n))$$

and covariance matrix

$$\begin{pmatrix} K(x_1, x_1) & \cdots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \cdots & K(x_n, x_n) \end{pmatrix}.$$

One particular benefit of this approach is that it yields both an estimate of the function (the mean vector) and an uncertainty quantification (the covariance matrix). So far, this choice to model f as a Gaussian Process does not use the observations $f(x_1), \ldots, f(x_n)$. However, when we have evaluated f at a few points, we can calculate the distribution of f conditional on these measurements. At any point $x \in \Omega$, the model then yields

$$f(x)|f(x_1), \ldots, f(x_n) \sim \mathcal{N}(\mu_*, K_*).$$

Here K_* only depends on the choice of the function K as well as the points $x_1, \ldots, x_n$ and x. μ_* depends on the choice of the function μ, the same points and the function values $f(x_1), \ldots, f(x_n)$ at these points (See (2) and (3)). We then have a model of f based on the known information. In Subsection 2.1, this conditional distribution is derived in detail.

In step 3, we use the selection criterion to find a promising next point $x_{n+1} \in \Omega$ at which we will evaluate f. Standard selection criteria for Bayesian Optimisation are optimising heuristics called acquisition functions. As an example, one of the most simple acquisition functions is the lower confidence bound

$$LCB(x) := \mu_*(x) - \xi K_*(x).$$

This acquisition function illustrates how both the function estimate $\mu_*(x)$ and the uncertainty quantification $K_*(x)$ are used. If we seek to minimise $LCB(x)$, we can

favour a point x for two reasons: firstly, if the function estimate is low, and secondly, if the uncertainty is high. As such, we encounter a trade-off between exploitation and exploration. This trade-off can be tuned by the **trade-off parameter** ξ; higher values of ξ value exploration more, and vice versa. We will discuss two other often-used acquisition functions in Subsection 2.3.
Lastly, in step 4 we continue this optimisation process by repeating steps 2 and 3. Each time, after step 3 we add the point x_{n+1} and associated function value $f(x_{n+1})$ to the known data $\mathcal{D}$, then model f using this updated data. This is repeated until we encounter a stopping criterion, which can for example be a maximum number of iterations or a threshold for the minimal value of f.

2.3. Acquisition functions

When the posterior distribution $f(x)|f(x_1), \ldots, f(x_n) \sim \mathcal{N}(\mu_*, K_*)$ has been derived, an acquisition function is used to decide where to sample next. Three of the most popular choices for this acquisition function are the lower confidence bound, the probability of improvement

$$\mathrm{PI}(x) := \mathbb{P}\left(f(x) \leq f^*\right)$$

and the expected improvement

$$\mathrm{EI}(x) := \mathbb{E}\left[\min(f(x) - f^*, 0)\right],$$

where f^* is the current minimal function value (over all points where the function has been evaluated). These acquisition functions have the benefit of weighing both the average $\mu(x)$ and the uncertainty $\sigma(x)$ in choosing where to evaluate next. We can also tweak this trade-off to favour either exploration or exploitation. In the case of the lower confidence bound, we achieve this by replacing the acquisition function by $\mu(x) - \alpha\sigma(x)$ for some $\alpha > 0$. As α gets larger, more uncertainty becomes more favourable and the method will favour exploration. Similarly, for the probability of improvement and expected improvement we can replace f^* by ξf^* for some $\xi > 0$. As this ξ gets larger, it will be as if the best found value increases. Critically, as ξ surpasses 1, there will be a gap between the actual best found value f^* and the best found value ξf^* the acquisition functions assume. As a result, the method will favor exploration. In the rest of this thesis, this ξ will be referred to as the **trade-off parameter**. The acquisition functions have the benefit of being less expensive to evaluate and having relatively cheap derivatives, so we can use a first- or second-order optimisation algorithm to minimise them.

For standard Bayesian optimisation, the expected improvement can actually be rewritten by using partial integration [Qin et al., 2017]. Simplify notation by forgetting x for the moment: we use $y = f(x)$ as our integration variable and write $\sigma := \sigma(x)$, $\mu := \mu(x)$. Then we have

$$
\begin{aligned}
\mathrm{EI} &= \int_{-\infty}^{\infty} \min(y - f^*, 0) \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(- \frac{(y-\mu)^2}{2\sigma^2} \Big) \, \mathrm{d}y \\
&= \int_{-\infty}^{f^*} (y - f^*) \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(- \frac{(y-\mu)^2}{2\sigma^2} \Big) \, \mathrm{d}y \\
&= \int_{-\infty}^{f^*-\mu} (y + \mu - f^*) \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(- \frac{y^2}{2\sigma^2} \Big) \, \mathrm{d}y \quad (y \mapsto y - \mu) \\
&= \int_{-\infty}^{(f^*-\mu)/\sigma} (\sigma y + \mu - f^*) \frac{1}{\sqrt{2\pi}} \exp\Big(- \frac{y^2}{2} \Big) \, \mathrm{d}y \quad (y \mapsto \frac{y}{\sigma}) \\
&= -\sigma \int_{-\infty}^{(f^*-\mu)/\sigma} \frac{\mathrm{d}}{\mathrm{d}y} \left[\frac{1}{\sqrt{2\pi}} \exp\Big(- \frac{y^2}{2} \Big) \right] \mathrm{d}y + (\mu - f^*) \Phi\left(\frac{f^* - \mu}{\sigma} \right) \\
&= (\mu - f^*) \Phi\left(\frac{f^* - \mu}{\sigma} \right) - \sigma \phi\left(\frac{f^* - \mu}{\sigma} \right),
\end{aligned}
$$

where Φ and ϕ are the cumulative distribution function and density function of a standard normal random variable, respectively.

Example 2.3 (optimising acquisition functions)**.** In Figure 2.3 we return to our earlier example and highlight three points where three acquisition functions are optimised.

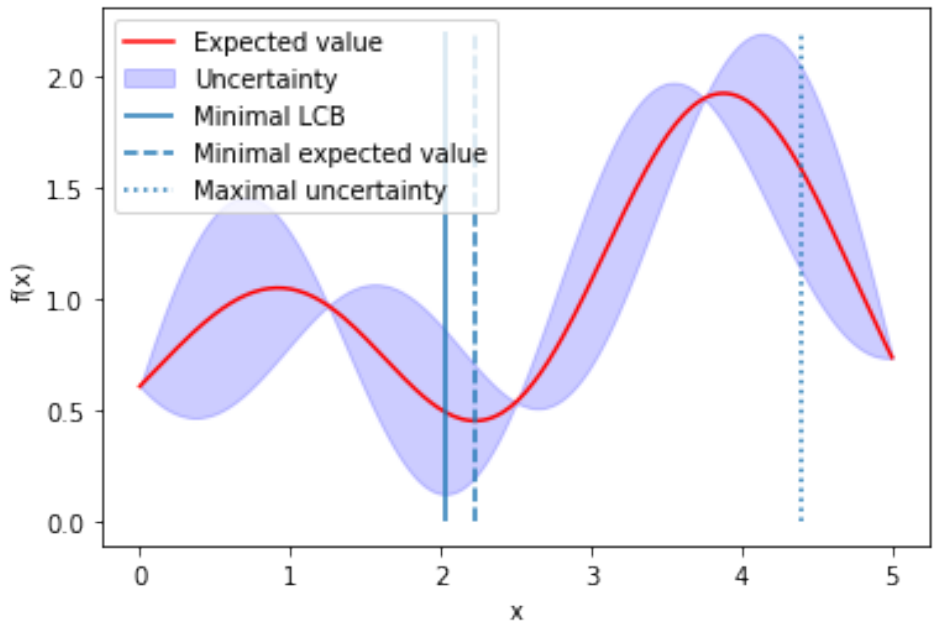

Figure 2.2: The points where a few simple acquisition functions are optimised.

For simplicity, the target function and measurement points have been omitted in the above figure. The red curve is the expected value $\mu_*(x)$ and the blue area is the '95

% certainty interval' enclosed by the curves $\mu_*(x) + 2\sigma_*(x)$ and $\mu_*(x) - 2\sigma_*(x)$, where $\sigma_*(x) := \sqrt{K_*(x,x)}$ is the posterior standard deviation. The vertical lines represent the optimal point to sample next according to three acquisition functions. The dashed line is where the expected value is minimised. Using this acquisition function leads to a 'pure exploitation' strategy, where the uncertainty quantification is not utilised. The dotted line is one of the points where this uncertainty quantification (the posterior standard deviation) is maximised. Similarly, this leads to a 'pure exploration' strategy, where the function estimate is not utilised. Combining these two acquisition functions brings us back to the lower confidence bound

$$\mathrm{LCB}(x) = \mu_*(x) - 2\sigma_*(x),$$

(here with a trade-off parameter $\xi = 2$), that lies in between the former two strategies, incorporating both exploitation and exploration. This last acquisition function is minimised at the solid line.

In the next subsection, we will discuss theoretical results that show how certain kernel functions guarantee desirable properties for the resulting covariance matrix.

2.4. Kernel functions

In this subsection, we will investigate properties of the covariance matrices generated by different kernel functions. For our purposes, there are two things we want to achieve. Firstly, the resulting matrix has to be invertible. Secondly, we want to be able to control the level of sparsity of the matrix. In particular, sparse matrices are computationally cheaper to invert.
The covariance matrix of a non-degenerate multivariate Gaussian is symmetric and positive definite. Bochner's theorem gives a necessary and sufficient condition for positive-definiteness of a matrix constructed by a kernel function. The theorem is very general, so here we specify an induced lemma for our use case.

Lemma 2.1 (P(S)D kernels)**.** *Let $\{x_1, \dots, x_n\} \subset \mathbb{R}^d$ be a set of n distinct points. A kernel matrix K with entries given by*

$$K_{ij} := g(x_i - x_j) \quad (1 \le i, j \le n)$$

is positive semi-definite if and only if the d-dimensional Fourier transform

$$\hat{g}(\xi) := \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} g(x) exp\Big(- i\xi \cdot x\Big) dx$$

of g is non-negative. Here, $\xi \cdot x$ denotes the inner product. Furthermore, $K_{1:n}$ is positive definite if g is non-negative and positive on a set $\Omega \subset \mathbb{R}^d$ of Lebesgue measure $\mu_L(\Omega) > 0$.

Proof. Let $c \in \mathbb{R}^n$ be any non-zero vector. What we have to show is that $c^\top K_{1:n} c \ge 0$ (or $c^\top K_{1:n} c > 0$ if we want to show that $K_{1:n}$ is positive definite). A straightforward calculation shows that

$$
\begin{aligned}
c^\top K_{1:n} c &= \sum_{k,l=1}^{n} c_k c_l g(x_k - x_l) \\
&= \sum_{k,l=1}^{n} c_k c_l \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \hat{g}(\xi) \exp\Big(i\xi \cdot (x_k - x_l)\Big) d\xi \\
&= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \hat{g}(\xi) \left(\sum_{k=1}^{n} c_k \exp\Big(i\xi \cdot x_k\Big) \right) \left(\sum_{l=1}^{n} c_l \exp\Big(-i\xi \cdot x_l\Big) \right) d\xi \\
&= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \hat{g}(\xi) \left| \sum_{k=1}^{n} c_k \exp\Big(i\xi \cdot x_k\Big) \right|^2 d\xi.
\end{aligned}
$$

Hence, $c^\top K_{1:n} c \geq 0$ for any non-zero $c \in \mathbb{R}^n$ if and only if the Fourier Transform $\hat{g}(\xi)$ is non-negative. Additionally, since we assumed $x_k \neq x_l$ for $k \neq l$ the functions given by

$$
\xi \mapsto \exp\Big(i\xi \cdot x_k\Big)
$$

are independent. Now let

$$
\mathcal{N} := \left\{ \xi \in \mathbb{R}^d \middle| h(\xi^*) := \sum_{k=1}^{n} c_k \exp\Big(i\xi \cdot x_k\Big) = 0 \right\}.
$$

We will prove by contradiction that this set has Lebesgue measure $\mu_L(\mathcal{N}) = 0$. Suppose the Lebesgue measure is positive; then a point ξ^* and a radius $r > 0$ exist such that $B(\xi^*, r)$, the open ball of radius r around ξ^*, is contained in $\mathcal{N}$. Now let $\psi \in B(\xi^*, r) \setminus \{\xi^*\}$ and define the function $g : \mathbb{R} \to \mathbb{C}$ by $g(x) = h(x\xi^* + (1-x)\psi)$. Note that g is an analytic function. Then $g(z) = 0$ for all $z \in (0, 1)$, and hence, by the identity theorem for complex-valued functions [Burdick and Lesley, 1975], g is the zero function. Since the choice of ϕ was arbitrary in $B(\xi^*, r)$, this implies that h is identically zero on $\mathbb{R}^d$, but this contradicts the linear independence of the exponentials.
To conclude, assume a set $\Omega \subset \mathbb{R}^d$ of positive Lebesgue measure exists on which $\hat{g}$ is positive. Then $\Omega \setminus \mathcal{N}$ also has positive Lebesgue measure, and as a result, under this assumption the integral above is strictly positive, so the kernel matrix $K_{1:n}$ is positive definite. □

This raises the question: which stationary functions have non-negative (and positive on an open set) Fourier transform? A well-known example is the squared exponential kernel function $K(x, x')_{SE} := \exp\Big(-\frac{\|x-x'\|_2}{2\sigma^2}\Big)$, since its Fourier transform is a scaled version of itself. However, the downside is that this kernel function is itself always positive, so the resulting matrix will not be sparse. The next lemma provides a sequence of stationary functions with compact support to remedy this problem, mentioned in [Wendland, 1995].

Lemma 2.2. *Let the* ***truncated power function*** $\psi_l : \mathbb{R}^d \to [0, \infty)$ *be given by*

$$\psi_l(x) := (1 - ||x||_2)_+^l,$$

where

$$(a)_+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then the d-dimensional Fourier transform

$$\hat{\psi}_l(\xi) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \psi_l(x) exp\Big(- i\xi \cdot x\Big) dx$$

is non-negative ***if*** $l \geq \lfloor d/2 \rfloor + 1$. *Furthermore,* $\hat{\psi}_l(\xi) \neq 0$ *for every* $\xi \neq 0 \in \mathbb{R}^d$.

We will prove the lemma for the case $d = 1$. The general proof is outside the scope of this report.

Proof. ($d = 1$ case) here we look at the series of functions

$$\psi_l : \mathbb{R} \ni x \mapsto (1 - |x|)_+^l \in [0, \infty).$$

We will prove that their scaled Fourier transform

$$\begin{aligned} \hat{\psi}_l(\xi) &= \int_{\mathbb{R}} \psi_l(x) \exp\Big(- i\xi x\Big) dx \\ &= \int_{-1}^{1} (1 - |x|)^l \exp\Big(- i\xi x\Big) dx \end{aligned}$$

is non-negative and that $\xi = 0$ is their only zero. The scaling (by a factor of $\sqrt{2\pi}$) is applied out of convenience, since it has no impact on the sign of the Fourier transform. The proof follows by induction. As our base case, for $l = 1$ we have

$$\hat{\psi}_1(\xi) = \int_{-1}^{1} \exp\Big(- i\xi x\Big) dx - \int_{-1}^{1} |x| \exp\Big(- i\xi x\Big) dx = \frac{2(1 - \cos \xi)}{\xi^2}.$$

From this expression we immediately see that

$$\hat{\psi}_1(\xi) \geq 0$$

is satisfied with equality only for $\xi = 0$. This concludes our base case.
For the induction step, we will start by proving that

$$\frac{d}{d\xi} \Big[\xi^{l+1} \hat{\psi}_l(\xi) \Big] = l \xi^l \hat{\psi}_{l-1}(\xi)$$

for all $l \geq 2$. We will use this equality later to show that the functions $f_l(\xi) := \xi^{l+1}\hat{\psi}_l(\xi)$ alternate between being odd and even and are strictly positive for $\xi > 0$, from which the desired property for $\hat{\psi}_l$ follows.
To prove the equality, we start by applying the product rule to the left-hand side. We find

$$\frac{d}{d\xi}\Big[\xi^{l+1}\hat{\psi}_l(\xi)\Big] = (l+1)\xi^l\hat{\psi}_l(\xi) + \xi^{l+1}\hat{\psi}_l'(\xi).$$

Thus, after dividing by ξ^l (the equality is obviously true for $\xi = 0$), the difference becomes

$$\frac{1}{\xi^l}\bigg(\frac{d}{d\xi}\Big[\xi^{l+1}\hat{\psi}_l(\xi)\Big] - l\xi^l\hat{\psi}_{l-1}(\xi)\bigg) = l(\hat{\psi}_l(\xi) - \hat{\psi}_{l-1}(\xi)) + \hat{\psi}_l(\xi) + \xi\hat{\psi}_l'(\xi).$$

Let's take a closer look at that first term. By using the fact that the anti-derivative of $l(1-|x|)^{l-1}$ is given by $\operatorname{sgn}(x)(1-|x|)^l$, we can rewrite it by partial integration. We find that

$$\begin{aligned}
l(\hat{\psi}_l(\xi) - \hat{\psi}_{l-1}(\xi)) &= -l\int_{-1}^{1} |x|(1-|x|)^{l-1}\exp\Big(-i\xi x\Big)dx \\
&= \Big[x(1-|x|)^l\exp\Big(-i\xi x\Big)\Big]_{x=-1}^{x=1} \\
&\quad - \int_{-1}^{1} \operatorname{sgn}(x)(1-|x|)^l\left(\operatorname{sgn}(x) - i\xi|x|\right)\exp\Big(-i\xi x\Big)dx \\
&= \xi\int_{-1}^{1}(1-|x|)^l ix\exp\Big(-i\xi x\Big)dx - \hat{\psi}_l(\xi).
\end{aligned}$$

Note that the first term on the last line is equal to $-\xi\hat{\psi}_l'(\xi)$. This can be seen by changing the order of differentiation and integration. In total, we thus see that the difference is

$$\frac{d}{d\xi}\Big[\xi^{l+1}\hat{\psi}_l(\xi)\Big] - l\xi^l\hat{\psi}_{l-1}(\xi) = \xi^l\Big(-\xi\hat{\psi}_l'(\xi) - \hat{\psi}_l(\xi) + \hat{\psi}_l(\xi) + \xi\hat{\psi}_l'(\xi)\Big) = 0.$$

Now we will show that the functions

$$f_l(\xi) = \xi^{l+1}\hat{\psi}_l(\xi),$$

indexed by $l \in \mathbb{N} \setminus \{0\}$, alternate between being odd (if l is even) and even (if l is odd) and are strictly positive for $\xi > 0$. We will prove this by induction as well. For our base case ($l = 1$), we have $f_1(\xi) = 2(1-\cos\xi)$, which is an even function and strictly positive for $\xi > 0$. For the induction step, first notice that $f_l(0) = 0$, since

$$\hat{\psi}_l(0) = \int_{-1}^{1}(1-|x|)^l dx = \frac{2}{l+1} < \infty.$$

Now we are interested in comparing $f_l(\xi)$ and $f_l(-\xi)$ for $\xi > 0$. The equality we proved previously reads

$$f_l'(\xi) = l f_{l-1}(\xi)$$

in terms of these functions. Then, by using the fundamental theorem of calculus, we find for $l \geq 2$ and $\xi > 0$ that

$$f_l(\xi) = f_l(\xi) - f_l(0) = \int_0^\xi f_l'(\chi) d\chi = l \int_0^\xi f_{l-1}(\chi) d\chi, \text{ and}$$
$$f_l(-\xi) = -(f_l(0) - f_l(-\xi)) = -\int_{-\xi}^0 f_l'(\chi) d\chi = -l \int_{-\xi}^0 f_{l-1}(\chi) d\chi.$$

Therefore, if f_{l-1} is an odd function, f_l is even and vice versa. Additionally, the expression for $f_l(\xi)$ shows that if $f_{l-1}(\xi) > 0$ for all $\xi > 0$, so is f_l.
Let's recap: what we have just proven is that the function

$$f_l(\xi) = \xi^{l+1} \hat{\psi}_l(\xi)$$

is strictly positive for all $\xi > 0$, and either odd (if l is even) or even (if l is odd). But then $\hat{\psi}_l(\xi)$ is also strictly positive for all $\xi > 0$ and always even, so it is actually strictly positive for all $\xi \neq 0$. This completes the proof. □

In general, if we have a radial function $\Phi(\cdot) = f(\|\cdot\|_2) : \mathbb{R}^d \to \mathbb{R}$, the d-dimensional Fourier transform can be reduced to a one-dimensional integral involving Bessel functions of the first kind. To achieve this, we start by switching from regular coordinates to polar coordinates:

$$\begin{aligned}\hat{\Phi}(\omega) &:= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \Phi(x) \exp\Big(- i\omega \cdot x \Big) dx \\ &= \frac{1}{(2\pi)^{d/2}} \int_0^\infty \int_0^\pi \dots \int_0^\pi \int_0^{2\pi} f(r) \exp\Big(- i\|\omega\|_2 r \cos\theta_{d-2} \Big) \\ &\cdot r^{d-1} \sin^{d-2}(\theta_{d-2}) \sin^{d-3}(\theta_{d-3}) \dots \sin\theta_1 d\theta_{d-1} \dots d\theta_1 dr.\end{aligned}$$

Here, the last line contains the volume element resulting from the coordinate transformation. Note that $\theta_1, \dots, \theta_{d-3}$ and θ_{d-1} only appear in said volume element. θ_{d-1} contributes a factor of 2π to the integral. For the other angular components, we have

$$\int_0^\pi \sin^k(\theta) d\theta = \sqrt{\pi} \frac{\Gamma((k+1)/2}{\Gamma((k+2)/2}, \quad k \geq 1.$$

Thus, the total contribution of $\theta_1, \dots, \theta_{d-3}$ is

$$\sqrt{\pi}\frac{\Gamma((d-2)/2)}{\Gamma((d-1)/2)} \cdot \sqrt{\pi}\frac{\Gamma((d-3)/2)}{\Gamma((d-2)/2)} \cdot \ldots \cdot \sqrt{\pi}\frac{\Gamma(1)}{\Gamma(3/2)}$$
$$= \pi^{(d-3)/2}\frac{\Gamma(1)}{\Gamma((d-1)/2)} = \frac{\pi^{(d-3)/2}}{\Gamma((d-1)/2)}.$$

Putting these contributions together, the Fourier transform becomes

$$\begin{aligned}\hat{\Phi}(\omega) &= \frac{1}{(2\pi)^{d/2}} \int_0^\infty \int_0^\pi f(r)\exp\Big(-i\|\omega\|_2 r\cos\theta_{d-2}\Big) \cdot r^{d-1}\sin^{d-2}(\theta_{d-2})2\pi\frac{\pi^{(d-3)/2}}{\Gamma((d-1)/2)}d\theta_{d-2}dr \\ &= \frac{1}{2^{(d-2)/2}\sqrt{\pi}\Gamma((d-1)/2)} \int_0^\infty r^{d-1}f(r)\int_0^\pi \exp\Big(-i\|\omega\|_2 r\cos\theta\Big)\sin^{d-2}(\theta)d\theta dr,\end{aligned}$$

where on the last line we dropped the subscript $d-2$ for the sole remaining angular component. Now, if we define the Bessel function of the first kind as

$$J_\nu(t) := \Big[2^\nu\sqrt{\pi}\Gamma(\nu+1/2)\Big]^{-1}t^\nu\int_0^\pi \exp\Big(-it\cos(\theta)\Big)\sin^{2\nu}(\theta)d\theta,$$

then we recognise this function in the Fourier Transform with $\nu = \frac{d-2}{2}$ and $t = \|\omega\|_2 r$. Plugging it in, we arrive at

$$\|\omega\|^{\frac{d-2}{2}}\hat{\Phi}(\omega) = \int_0^\infty r^{\frac{d-2}{2}}J_{\frac{d-2}{2}}(\|\omega\|_2 r)f(r)rdr. \tag{4}$$

To recap, what we have done is simplified the d-dimensional Fourier transform to a 1-dimensional integral, albeit an integral involving the Bessel function of the first kind, which in general has no closed expressions. Providing a proof that the truncated power functions given in theorem 2.2 have positive Fourier transform for general $d \geq 2$ is outside the scope of this thesis.

3. Bayesian Optimisation for Vector-Valued Functions

In the previous section, we discussed the framework of Bayesian Optimisation. In this section, the algorithm for $\|\cdot\|$-minimisation will be introduced. We will start by deriving the distribution of the largest absolute value of multiple Gaussians, which will be used to model the components of our function. This distribution directly yields the probability of improvement acquisition function, and we also use it to simplify the expected improvement acquisition function. With this out of the way, we give an overview of the algorithm.

3.1. Problem definition

Our interest is in an optimisation problem

$$\min_{x\in\Omega} g(f(x)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^m$ is an unknown function, $g : \mathbb{R}^m \to \mathbb{R}$, given by

$$g(y) = \|y\|_\infty,$$

is the maximum absolute value norm and Ω is a compact subset of $\mathbb{R}^d$. The approach we suggest for solving this problem involves applying Gaussian Process Regression independently to each component of f. For illustration, we will first discuss the case where $m = 1$.

Example 3.1 (Optimisation problem for $m = 1$)**.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be some unknown function. Our optimisation problem now becomes

$$\min_{x\in\Omega} |f(x)|,$$

where Ω is a compact subset of $\mathbb{R}^d$. Given observations $\mathcal{D} = \{x_1, \ldots, x_n, f(x_1), \ldots, f(x_n)\}$, our approach is to use Bayesian Optimisation. The model assumption is that f is a Gaussian Process with mean function $\mu : \mathbb{R}^d \to \mathbb{R}$ and kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. We now propose a new acquisition function

$$\mathbb{P}\left(|f(x)| \leq f^*\right),$$

where $f^* = \min_{x\in\mathcal{D}} |f(x)|$ is the best found value so far. This acquisition function is promising because optimising it will yield the next sample with the highest probability of being an improvement under the model assumptions of Gaussian Process Regression.

For $m > 1$, we need a different approach in a Bayesian Optimisation framework. The function $f : \mathbb{R}^d \to \mathbb{R}^m$ will have multiple outputs. One approach would be to do Bayesian Optimisation on the composite function $g \circ f$. However, this approach does not utilise the information given by the outputs of the components of f. The approach

we suggest starts by modelling each of these components $f_1, \ldots, f_m$ as an independent Gaussian Process. Then we define new selection criteria based on acquisition functions that rely on each of these Gaussian Processes. Given observations

$$\mathcal{D} = \{x_1, \ldots, x_n, (f_1(x_1), \ldots, f_m(x_1)), \ldots, (f_1(x_n), \ldots, f_m(x_n))\},$$

the approach is to select as the next sampling location the point x that maximises the **probability of improvement**

$$\mathrm{PI}(x) := \mathbb{P}\left(\|f(x)\| \leq f^*\right),$$

where $f^* = \min_{x \in \mathcal{D}} \|f(x)\|_\infty$. To maximise this function, we need to study the random variable $\|f(x)\|_\infty$, i.e. the maximum absolute value of independent Gaussians. In the next subsection, we will derive an analytical formula for the distribution of this random variable in terms of the cumulative distribution function of the standard normal random variable and the mean and kernel functions of the individual components of f.

3.2. Distribution of the maximum norm of independent Gaussians

We start with deriving the distribution of

$$Y := \max_{1 \leq i \leq n} |X_i|,$$

the maximum absolute value of n independent random normal variables $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$.

To achieve this, we use the folded normal distribution, which tells us that if $X \sim \mathcal{N}(\mu, \sigma)$, then $|X|$ has density

$$f_{|X|}(z) = \frac{1}{\sigma\sqrt{2\pi}}\left(e^{-\frac{(z-\mu)^2}{2\sigma^2}} + e^{-\frac{(z+\mu)^2}{2\sigma^2}}\right) \tag{5}$$

for $z \geq 0$, and 0 for $z < 0$. For $z \geq 0$ this is the sum of the densities of two normal random variables with variance σ^2, where one of them is centred around μ and the other around $-\mu$. This result can be derived by differentiating the cumulative distribution function, given by

$$\mathbb{P}(|X| \leq z) = \mathbb{P}(-z \leq X \leq z) = \int_{-z}^{z} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}\, \mathrm{d}t.$$

Before we write down the cumulative distribution function for Y, we first rewrite it for $|X|$ in terms of

$$\Phi(z) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{1}{2}t^2}\, \mathrm{d}t,$$

the cumulative distribution function of a standard normal random variable. The integral of the first part of (5) can then be expressed as

$$\begin{aligned}\int_0^z \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}\,\mathrm{d}t &= \int_{-\mu}^{z-\mu} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}}\,\mathrm{d}t && (t \mapsto t-\mu)\\ &= \int_{-\mu/\sigma}^{(z-\mu)/\sigma} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}\,\mathrm{d}t && (t \mapsto t/\sigma)\\ &= \Phi\left(\frac{z-\mu}{\sigma}\right) - \Phi\left(\frac{-\mu}{\sigma}\right).\end{aligned}$$

If we replace $-\mu$ by μ, we immediately find the corresponding expression for the second part of (5) to be

$$\int_0^z \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t+\mu)^2}{2\sigma^2}}\,\mathrm{d}t = \Phi\left(\frac{z+\mu}{\sigma}\right) - \Phi\left(\frac{\mu}{\sigma}\right).$$

By adding them together, we find that

$$\begin{aligned}\mathbb{P}(|X| \le z) &= \Phi\left(\frac{z-\mu}{\sigma}\right) - \Phi\left(\frac{-\mu}{\sigma}\right) + \Phi\left(\frac{z+\mu}{\sigma}\right) - \Phi\left(\frac{\mu}{\sigma}\right)\\ &= \Phi\left(\frac{z-\mu}{\sigma}\right) + \Phi\left(\frac{z+\mu}{\sigma}\right) - 1, \qquad (6)\end{aligned}$$

where we used the fact that Φ has the symmetry $\Phi(z) = 1 - \Phi(-z)$. The fact that we subtract 1 here can be interpreted as removing the probability of $\frac{1}{2}$ that a standard normal random variable is negative twice.

We can now use a common trick to write down the cumulative distribution function for $Y := \max_{1\le i\le n} |X_i|$: the maximum of a number of random variables is smaller than some value $z > 0$ if and only if all of them are smaller than z. Therefore, and since we assume $X_1, \ldots, X_n$ to be independent, we have

$$\begin{aligned}\mathbb{P}(Y \le z) &= \prod_{i=1}^{n} \mathbb{P}(X_i \le z)\\ &= \prod_{i=1}^{n} \left[\Phi\left(\frac{z-\mu_i}{\sigma_i}\right) + \Phi\left(\frac{z+\mu_i}{\sigma_i}\right) - 1\right]. \qquad (7)\end{aligned}$$

Now that we have this cumulative distribution function, we can introduce a new acquisition function: the probability of improvement for maximum absolute value minimisation. In our Bayesian Optimisation model, the random variable $\|f(x)\|_\infty$ is distributed similarly to Y above. In fact, if we replace μ_i by $\mu_i(x)$ and σ_i by $\sigma_i(x)$ in

Equation (7), i.e. the posterior mean and standard deviation of function component f_i at point $x \in \Omega$, it spells out the distribution of $\|f(x)\|_\infty$. As a result, the probability of improvement for maximum absolute value is given by

$$\mathrm{PI}(x) := \mathbb{P}(\|f(x)\| \leq f^*) = \prod_{i=1}^{n} \left[\Phi\left(\frac{f^* - \mu_i(x)}{\sigma_i(x)} \right) + \Phi\left(\frac{f^* + \mu_i(x)}{\sigma_i(x)} \right) - 1 \right]. \tag{8}$$

.

In the next subsection, we introduce an alternative acquisition function: the expected improvement for maximum absolute value minimisation.

3.3. An alternative acquisition function based on expected improvement

Next to the probability of improvement for max-abs minimisation, we propose an alternative acquisition function: the expected improvement for max-abs minimisation. To recap, our goal is to minimise $\|f\|_\infty$, where f is a vector-valued function from $\Omega \subset \mathbb{R}^d$ to $\mathbb{R}^m$. First we use Gaussian Process Regression on each component individually, to model their output at a point $x \in \Omega$ as $f_i(x) \sim \mathcal{N}(\mu_i(x), \sigma_i^2(x))$ $(1 \leq i \leq m)$. Suppose we have measured n data points $x_1, \ldots, x_n$ so far, and set

$$f^* := \min_{1 \leq i \leq n} \|f(x_i)\|_\infty$$

to be the best current measurement. Then the expected improvement is given by

$$EI(x) = \mathbb{E}\left[\max(f^* - \|f(x)\|_\infty, 0)\right], \tag{9}$$

where the distribution of $\|f(x)\|_\infty$ is given by equation (7) as derived in section 3.2. Our selection criterion is finding the point $x \in \Omega$ that maximises this expected improvement. Theorem 3.1 tells us how we can calculate this expected improvement.

Theorem 3.1. *The expected improvement* (9) *is equal to*

$$\int_0^{f^*} \mathbb{P}\left(\|f(x)\|_\infty \leq y\right) \mathrm{d}y.$$

We prove this result in two ways. Below is the shorter proof, which relies on the derived distribution of $\|f(x)\|$. A more convoluted proof, writing the expected improvement as an m-dimensional integral and using induction, can be found in Appendix A.2.

Proof. Write Ψ and ψ for (7) and the derivative of (7), respectively, and note that these functions only have support on the positive real line. These functions depend on x through $\mu(x)$ and $\sigma(x)$, but we omit this dependence here to simplify notation. Then we find

$$\begin{aligned}\mathrm{EI} &= \int_{-\infty}^{\infty} \max(f^* - y, 0)\psi(y)\,\mathrm{d}y \\ &= \int_{0}^{f^*} (f^* - y)\psi(y)\,\mathrm{d}y \\ &= f^*\Psi(f^*) - \int_{0}^{f^*} y\psi(y)\,\mathrm{d}y.\end{aligned}$$

Now apply partial integration to this integral to find that

$$\mathrm{EI} = f^*\Psi(f^*) - f^*\Psi(f^*) + \int_{0}^{f^*} \Psi(y)\,\mathrm{d}y = \int_{0}^{f^*} \Psi(y)\,\mathrm{d}y.$$

□

4. Global optimisation of acquisition functions in single-output Bayesian Optimisation

Bayesian Optimisation is a sequential method to solve optimisation problems, in which the selection of the next sampling point relies on the optimisation of an acquisition function (see Section 2). In the previous section, we introduced two new acquisition functions that allow us to extend Bayesian Optimisation to the multivariate optimisation problem

$$\min_{x\in\Omega} \|f(x)\|_\infty,$$

where $\Omega \subset \mathbb{R}^d$ is a bounded domain, $f : \mathbb{R}^d \to \mathbb{R}^m$ is an unknown function and $\|\cdot\|_\infty$ is the maximum absolute-value norm. However, the work is not done when we have derived a formula for an acquisition function. As it turns out, most acquisition functions are non-convex and suffer from a lot of plateaus of constant value. This last part grows increasingly troublesome as the input dimension of the problem increases. Since the acquisition function at some point in the input space $x \in \Omega$ depends on x only through the mean and variance predictions $\mu(x)$ and $\sigma^2(x)$, the function will be constant on some subdomain Ω' if these predictions are constant there. As we use the prior assumption $\mu_0(x) = 0$, this being constant of the predictions on subdomains is unavoidable as the dimension increases **unless** we are willing to allow the characteristic lengthscale to grow large, but doing so brings a complexity penalty to the optimisation problem, as well as the possible cost of the lengthscale not being representative for the model of the function.

In order to find the global optimum of the acquisition function, we propose a branch-and-bound algorithm. See [Fowkes et al., 2013] for an algorithm that minimises functions with a Lipschitz-continuous Hessian. In this context, we will not make these assumptions, but instead apply a similar method that only requires the Lipschitz-continuity of the acquisition function itself. This simplification has a downside: the bounds we apply for optimising the acquisition functions will be less tight. Therefore, investigating the Lipschitz constant of the gradient and Hessian of acquisition functions is left as an avenue for future research.
In this section, we will discuss Lipschitz-continuity for some acquisition functions in the $m = 1$ regime; that is, we consider the case of applying Bayesian Optimisation to single-output functions $f : \mathbb{R}^d \to \mathbb{R}$. That is, for a given acquisition function $a : \mathbb{R}^d \supset \Omega \to \mathbb{R}$ we will derive the Lipschitz constant $\alpha_a > 0$ such that

$$|a(x) - a(y)| \leq \alpha_a \|x - y\|_2,$$

where $\|\cdot\|_2$ denotes the Euclidean norm on the input space $\Omega \subset \mathbb{R}^d$. Armed with this constant, we will be able to systematically disregard sections of the input space Ω during the optimisation of a.

Example 4.1 (branch-and-bound optimisation)**.** Suppose we have a function $h : [0, 1] \to \mathbb{R}$ for which the Lipschitz constant is $\alpha_h = 1$, which we want to maximise. We

have measured h at $x = 0.25$ and $x = 0.75$, with results $h(0.25) = 0.25$ and $h(0.75) = 0$. In figure 4.1, we see the upper bound we find by combining the Lipschitz-continuity and the known function values. If we want to maximise this function, we can now disregard the subdomain $[0.5, 1]$. This is because on $[0.5, 1]$ the function cannot yield a value higher than 0.25, which we already found at $x = 0.25$.

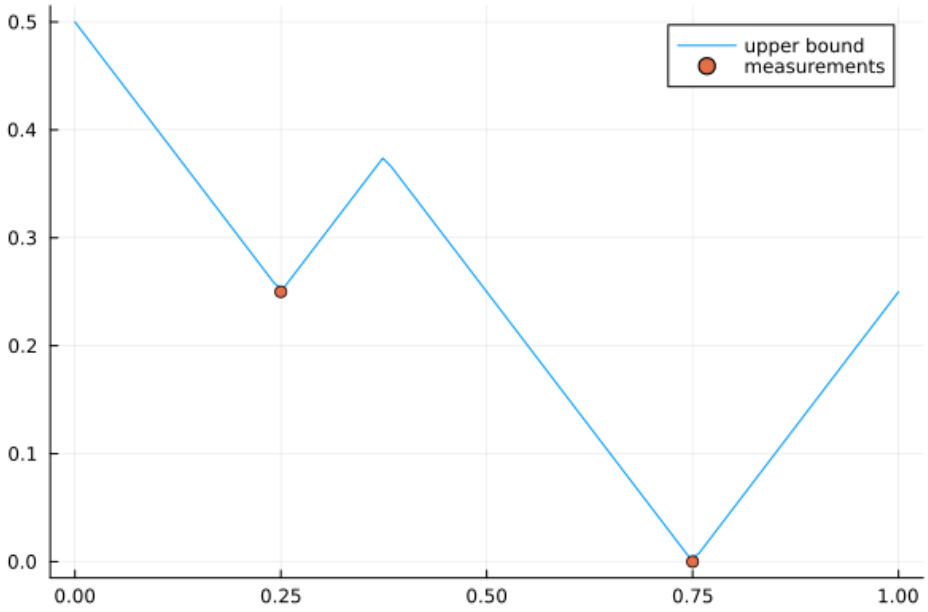

Figure 4.1: The Lipschitz upper bound on an unknown function h with Lipschitz constant $\alpha_h = 1$.

4.1. Lipschitz-continuity of Gaussian Process Regression

In this subsection, we will derive Lipschitz-continuity for the posterior mean and variance, given by

$$\mu_*(y) = \mu(y) + K(y, x_{1:n})K_{1:n}^{-1}(f_{1:n} - \mu_{1:n}) = K(y, x_{1:n})K_{1:n}^{-1}f_{1:n} \text{ and}$$
$$v_*(y) = K(y, y) - K(y, x_{1:n})K_{1:n}^{-1}K(x_{1:n}, y) = 1 - K(y, x_{1:n})K_{1:n}^{-1}K(x_{1:n}, y),$$

as shown in equations (2) and (3). Here $x_{1:n}$ denotes the vector $(x_1, \ldots, x_n)$ of inputs we have sampled so far, and likewise $f_{1:n}$ denotes the vector $(f(x_1), \ldots, f(x_n))$ containing the resulting samples. The latter two identities are results of specific choices for the functions that generate the Gaussian Process: for our mean function we choose the zero function $\mu(x) = 0$, and for our kernel function we consider two options: the truncated power kernel $K(x, y) = (1 - \|x - y\|_2)_+^p$, where $(a)_+ = \max(a, 0)$, and the squared exponential kernel $K(x, y) = \exp\left(-\frac{\|x-z\|_2^2}{2\ell^2}\right)$. We proceed in two steps: first we

derive Lipschitz constants for both these kernel functions with one input fixed, then we build on this result to derive the Lipschitz constant for the posterior mean and variance of the Gaussian process. In the next subsection, we will propose modified acquisition functions based on (i) the lower confidence bound, (ii) the probability of improvement and (iii) the expected improvement acquisition functions; these modified acquisition functions are universally Lipschitz-continuous. As a first step towards obtaining this result, we'll start by demonstrating the Lipschitz-continuity of both kernel functions with one fixed input.

Lemma 4.1 (Lipschitz-continuity of the kernel functions)**.** *Let $F_z : \mathbb{R}^d \to \mathbb{R}$ be given by $F_z(x) = K(x,z) = (1 - \|x - z\|_2)_+^p$. Then F_z is Lipschitz-continuous with constant $\alpha_K = p$, independent of the value of $z \in \mathbb{R}^d$. That is,*

$$\left|F_z(x) - F_z(y)\right| \leq p\|x - y\|_2 \quad \forall x, y, z \in \mathbb{R}^d.$$

If F_z is given by $F_z(x) = K(x,z) = exp\Big(-\frac{\|x-z\|_2^2}{2\ell^2}\Big)$, then it is Lipschitz-continuous with constant $\alpha_K = \frac{1}{\ell\sqrt{e}}$, independent of the value of $z \in \mathbb{R}^d$.

Proof. To start, let $h_z(x) := \|x - z\|_2$. Then h_z is Lipschitz-continuous with constant $\alpha_h = 1$ by the inverse triangle inequality:

$$\left|\|x - z\|_2 - \|y - z\|_2\right| \leq \left\|(x - z) - (y - z)\right\|_2 = \|x - y\|_2.$$

Now, in case of the truncated power kernel function, we can express F_z as $F_z = g \circ h_z$, where $g : \mathbb{R} \to \mathbb{R}$ is given by

$$g(x) = (1 - x)_+^p.$$

Since g is differentiable on $\mathbb{R}$, the mean value theorem states that

$$\forall x, y \in \mathbb{R} \quad \exists c \in R : g(x) - g(y) = g'(c)(x - y).$$

Therefore, g is Lipschitz-continuous with constant

$$\alpha_g = \sup_{c \in \mathbb{R}} \left|g'(c)\right| = \sup_{c \in [0,1]} \left|p(1 - c)^{p-1}\right| = p.$$

Now we can put these results together to see that

$$\left|F_z(x) - F_z(y)\right| = \left|g(h_z(x)) - g(h_z(y))\right| \leq p\left|h_z(x) - h_z(y)\right| \leq p\|x - y\|_2.$$

This proves the Lipschitz-continuity of the truncated power kernel function. In case of the squared exponential kernel function, we can choose

$$g = \exp\Big(-\frac{x^2}{2\ell^2}\Big).$$

This function has a bounded derivative with supremum

$$\sup_{x \in \mathbb{R}} \left|g'(x)\right| = \frac{1}{\ell\sqrt{e}}.$$

Hence, by a similar argument it follows that the squared exponential kernel function with one fixed input is Lipschitz-continuous with constant $\alpha_K = \frac{1}{\ell\sqrt{e}}$. □

Using this result, we can show that the posterior mean and variance are Lipschitz-continuous.

Lemma 4.2 (Lipschitz-continuity of the posterior distribution)**.** *Let the posterior distribution be given by*

$$\begin{aligned}\mu_*(y) &= K(y, x_{1:n})K_{1:n}^{-1}f_{1:n} \text{ and}\\ v_*(y) &= 1 - K(y, x_{1:n})K_{1:n}^{-1}K(x_{1:n}, y).\end{aligned}$$

Then μ_* is Lipschitz-continuous with constant

$$\alpha_\mu = \sqrt{n}\left\|K_{1:n}^{-1}f_{1:n}\right\|_2 \alpha_K$$

and v_* is Lipschitz-continuous with constant

$$\alpha_v = n\frac{2}{\lambda_{min}(K_{1:n})}\alpha_K,$$

where $\lambda_{min}(A)$ denotes the smallest eigenvalue of a positive definite matrix A. Here α_K is the Lipschitz constant for the kernel function with one fixed input; either p for the truncated power kernel function or $\frac{1}{\ell\sqrt{e}}$ for the squared exponential kernel function.

Proof. First, the 2-norm of the vector-valued function $y \mapsto K(y, x_{1:n})$ is Lipschitz-continuous with constant $\sqrt{n}\alpha_K$. To see this, note that

$$\begin{aligned}\left\|K(y, x_{1:n}) - K(z, x_{1:n})\right\|_2^2 &= \sum_{i=1}^{n}\left(K(y, x_i) - K(z, x_i)\right)^2\\ &\leq \sum_{i=1}^{n}\alpha_K^2\|y - z\|_2^2 = n\alpha_K^2\|y - z\|_2^2.\end{aligned}$$

Taking the square root on both sides of the inequality yields the desired result. Then we can straightforwardly apply the Cauchy-Schwarz inequality to derive the Lipschitz-continuity and an upper bound for the Lipschitz constant of the posterior mean:

$$\begin{aligned}|\mu_*(y) - \mu_*(z)| &= \left|\left(K(y, x_{1:n}) - K(z, x_{1:n})\right)K_{1:n}^{-1}f_{1:n}\right|\\ &\leq \left\|K_{1:n}^{-1}f_{1:n}\right\|_2\left\|K(y, x_{1:n}) - K(z, x_{1:n})\right\|_2\\ &\leq \sqrt{n}\left\|K_{1:n}^{-1}f_{1:n}\right\|_2\alpha_K\|y - z\|_2.\end{aligned}$$

For the posterior variance v_*, we find

$$
\begin{aligned}
|v_*(y) - v_*(z)| &= \left|K(y, x_{1:n})K_{1:n}^{-1}K(x_{1:n}, y) - K(z, x_{1:n})K_{1:n}^{-1}K(x_{1:n}, z)\right| \\
&= \left|\left[K(y, x_{1:n}) - K(z, x_{1:n})\right] K_{1:n}^{-1} \left[K(x_{1:n}, y) + K(x_{1:n}, z)\right]\right| \\
&\leq \left\|K(y, x_{1:n}) - K(z, x_{1:n})\right\|_2 \lambda_{\max}\left(K_{1:n}^{-1}\right) \left\|K(x_{1:n}, y) + K(x_{1:n}, z)\right\|_2 \\
&\leq \sqrt{n}\alpha_K \|y - z\|_2 \frac{1}{\lambda_{\min}(K_{1:n})} \sqrt{4n} \\
&= n\frac{2}{\lambda_{\min}(K_{1:n})}\alpha_K \|y - z\|_2,
\end{aligned}
$$

where in the last inequality we used two facts. Firstly, for a positive definite matrix A, the largest eigenvalue of the inverse A^{-1} is equal to the reciprocal of its smallest eigenvalue. Secondly, both the truncated power kernel function and the squared exponential kernel function have the range $[0, 1] \subset \mathbb{R}$. Therefore, we find that

$$
\left\|K(x_{1:n}, y) + K(x_{1:n}, z)\right\|_2^2 = \sum_{i=1}^{n} \left(K(x_i, y) + K(x_i, z)\right)^2 \leq \sum_{i=1}^{n} 2^2 = 4n.
$$

□

In the next subsection, we investigate three acquisition functions which are not Lipschitz-continuous. For each, we suggest modifications and derive the Lipschitz constant of the resulting modified acquisition function.

4.2. Lipschitz-continuous acquisition functions for single-output Bayesian Optimisation

In this subsection, we will consider three acquisition functions in the regime of Bayesian Optimisation of a single-output function $f : \Omega \to \mathbb{R}$, for some compact set $\Omega \subset \mathbb{R}^d$: (i) the lower confidence bound, (ii) the probability of improvement and (iii) the expected improvement, as introduced in Subsection 2.3. We will suggest some changes to make these acquisition functions Lipschitz-continuous when considered as functions from Ω to $\mathbb{R}$, respectively the input and output space of our optimisation problem. First, we consider the lower confidence bound. Traditionally, this acquisition function is defined as

$$
\mathrm{LCB}(x) := \mu_*(x) - \xi\sigma_*(x),
$$

where $\xi > 0$ is a trade-off parameter: higher values favour more exploration. However, this function is not Lipschitz-continuous in terms of the variance v_* due to σ_* being the root of v_*: as x tends to zero, the derivative of the square root function $f(x) = \sqrt{x}$ becomes unbounded. Therefore, we will investigate the effects of replacing this acquisition function by the **modified lower confidence bound**

$$
\mathrm{MLCB}(x) := \mu_*(x) - \xi v_*(x).
$$

This acquisition function is Lipschitz-continuous. A potential downside is that this function is non-physical: if we are optimising some physical problem, then μ_* and v_* have different dimensions. From a statistical perspective, this modified function also has less meaning: whereas for the normal distribution, the probability that a measurement of a random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ yields a result below $\mu - \sigma$ is approximately 0.16, this modified lower confidence bound has no such interpretation. These observations might imply that the adapted acquisition function has less predictive power, and is thus less suited to Bayesian Optimisation.
Next, we consider the probability of improvement acquisition function. It is a function of the posterior mean $\mu_*(x)$ and variance $v_*(x)$, where for simplicity we will omit the inner dependence on x. It is then given by

$$\mathrm{PI}(\mu_*, v_*) := \Phi\left(\frac{f^* - \mu_*}{\sqrt{v_*}}\right),$$

where f^* is the lowest function value found during Bayesian Optimisation so far and Φ is the cumulative distribution function of a standard normal random variable. Considered as a function $\mathrm{PI} : \mathbb{R} \times [0, \infty) \to [0, 1]$ of μ_* and v_*, this acquisition function is not globally Lipschitz. This is because

$$\lim_{v_* \downarrow 0} \Phi\left(\frac{f^* - \mu_*}{\sqrt{v_*}}\right) = \begin{cases} 0 & \text{if } f^* < \mu_*, \\ \frac{1}{2} & \text{if } f^* = \mu_*, \text{ and} \\ 1 & \text{if } f^* > \mu_*. \end{cases}$$

As such, the partial derivative of this function with respect to μ_* is unbounded around $(\mu_*, v_*) = (f^*, 0)$, and therefore not universally Lipschitz. One might still wonder whether these values for μ_* and v_* are ever attained simultaneously in Gaussian Process Regression. This is indeed the case when $x \in \Omega$ is close to x_*, a measured point at which the current minimum f^* was found, because as x approaches a measured point v_* tends to zero and μ_* approaches the function value at that measured point. One can check this using Equations (2) and (3). To remedy this problem, we propose the **modified probability of improvement**

$$\mathrm{MPI}_\epsilon(\mu_*, v_*) := \Phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}}\right)$$

for some $\epsilon > 0$. This is equivalent to adding the constant term ϵ to the posterior variance v_*. We expect this change will affect the effectiveness of the acquisition function by reducing the relative information carried by the posterior variance v_*. If, over the domain Ω, the range of the variance shifts from $[0, v_{\max}]$ (for some $v_{\max} > 0$) to $[\epsilon, v_{\max} + \epsilon]$, this might result in the relative differences in v_* being less informative. It should be noted that this is a rough prediction based on the intuition of the author. Investigating this result is outside the scope of this report, and is left as an avenue for future research.

Lastly, we turn our attention to the expected improvement acquisition function. As derived in Subsection 2.3, in the context of optimising a univariate function $f : \Omega \to \mathbb{R}$, this acquisition function is given by

$$\mathrm{EI}(\mu_*, v_*) := (\mu_* - f^*)\Phi\left(\frac{f^* - \mu_*}{\sqrt{v_*}}\right) - \sqrt{v_*}\phi\left(\frac{f^* - \mu_*}{\sqrt{v_*}}\right),$$

in terms of the best function value f^* found so far and the posterior mean $\mu_*(= \mu_*(x))$ and variance $v_*(= v_*(x))$. Similar to the probability of improvement, this acquisition function is not globally Lipschitz-continuous when considered as a function EI: $\mathbb{R} \times [0, \infty) \to \mathbb{R}$ of μ_* and v_*. In particular, the partial derivative with respect to v_* is unbounded around $(\mu_*, v_*) = (f^*, 0)$, as can be easily checked. Therefore, analogously to the modified probability of improvement, we propose the **modified expected improvement**

$$\mathrm{MEI}_\epsilon(\mu_*, v_*) := (\mu_* - f^*)\Phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}}\right) - \sqrt{v_*}\phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}}\right),$$

for some $\epsilon > 0$. Now, we can derive Lipschitz constants for these three modified acquisition functions.

Theorem 4.3. *Suppose we are in the process of minimising a univariate function $f : \Omega \to \mathbb{R}$ for some compact $\Omega \subset \mathbb{R}^d$. So far, we have sampled f at points $x_1, \ldots, x_n$ and stored the results in the vector $f_{1:n} := (f(x_1), \ldots, f(x_n))$. The lowest value found so far is denoted by $f^* := \min_{1 \le i \le n} f(x_i)$. The modified lower confidence bound, modified probability of improvement and modified expected improvement acquisition functions are then given by*

$$\begin{aligned}
MLCB(x) &:= \mu_*(x) - \xi v_*(x), \\
MPI_\epsilon(\mu_*, v_*) &:= \Phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}}\right) \text{ and} \\
MEI_\epsilon(\mu_*, v_*) &:= (\mu_* - f^*)\Phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}}\right) - \sqrt{v_*}\phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}}\right),
\end{aligned}$$

respectively, where Φ and ϕ denote the cumulative and density distribution functions of a standard normal random variable, respectively, μ_* is the posterior mean function, v_* is the posterior variance function, ξ is the exploration-exploitation trade-off parameter and ϵ is a positive real number added to the posterior variance for the MPI and MEI.
Then the modified lower confidence bound has Lipschitz constant

$$\alpha_{MLCB} = \left(\sqrt{n}\left\|K_{1:n}^{-1} f_{1:n}\right\|_2 + \xi n \frac{2}{\lambda_{min}(K_{1:n})}\right)\alpha_K,$$

the modified probability of improvement has Lipschitz constant

$$\alpha_{MPI} = \frac{1}{\sqrt{2\pi}}\left(\sqrt{\frac{n}{\epsilon}}\left\|K_{1:n}^{-1} f_{1:n}\right\|_2 + \frac{n}{2\epsilon}\frac{e^{-\frac{1}{2}}}{\lambda_{min}(K_{1:n})}\right)\alpha_K$$

and the modified expected improvement has Lipschitz constant

$$\alpha_{MEI} = \left(\sqrt{n} \left(1 + \frac{|f^*|}{\sqrt{2\pi}} \right) \left\| K_{1:n}^{-1} f_{1:n} \right\|_2 + \frac{n}{\sqrt{2\pi\epsilon}} \frac{1}{\lambda_{min}(K_{1:n})} \right) \alpha_K.$$

Here α_K is the Lipschitz constant for the kernel function with one fixed input; either p for the truncated power kernel function or $\frac{1}{\ell\sqrt{e}}$ for the squared exponential kernel function.

Proof. Recall the Lipschitz constants of the posterior mean μ_* and posterior variance v_*, considered as functions from Ω to $\mathbb{R}$, as given in Lemma 4.2. The Lipschitz constant for the modified lower confidence bound acquisition function then follows by straightforward application of the triangle inequality:

$$\begin{aligned} |\mathrm{MLCB}(x) - \mathrm{MLCB}(y)| &\leq |\mu_*(x) - \mu_*(y)| + \xi |v_*(x) - v_*(y)| \\ &\leq \left(\sqrt{n} \left\| K_{1:n}^{-1} f_{1:n} \right\|_2 + \xi n \frac{2}{\lambda_{\min}(K_{1:n})} \right) \alpha_K \|x - y\|_2 \end{aligned}$$

For the modified probability of improvement and modified expected improvement, we will also rely on the known Lipschitz constants of μ_* and v_*. A generalisation of the Mean Value Theorem to continuously differentiable functions $F : \mathbb{R}^2 \to \mathbb{R}$ of two variables tells us that

$$\forall u, v \in R^2 \exists c \in [0,1] : F(u) - F(v) = \nabla F\left((1-c)u + cv\right) \cdot (v - u),$$

where $\cdot$ denotes the standard inner product and ∇ denotes the gradient operator. To see this, one can define a function $g : [0,1] \to \mathbb{R}$ by $g(t) = F((1-t)u + tv)$ and apply the Mean Value Theorem to this function. In our setting, we use this result by replacing the vectors $u, v \in \mathbb{R}^2$ by $(\mu_*(x), v_*(x))$ and $(\mu_*(y), v_*(y))$ for some $x, y \in \Omega$. Then for any acquisition function a that only depends on $x \in \Omega$ through the posterior mean and variance, we can derive the following estimate:

$$\begin{aligned} |a(x) - a(y)| &= \left| a(\mu_*(x), v_*(x)) - a(\mu_*(y), v_*(y)) \right| \\ &\leq \sup_{\mu \in \mathbb{R}, v \in [0,\infty)} \left| \frac{\partial a}{\partial \mu}(\mu, v) \right| \left| \mu_*(x) - \mu_*(y) \right| \\ &+ \sup_{\mu \in \mathbb{R}, v \in [0,\infty)} \left| \frac{\partial a}{\partial v}(\mu, v) \right| \left| v_*(x) - v_*(y) \right| \end{aligned}$$

by explicitly writing out the terms in the inner product. Using the supremum of the partial derivatives is necessary here because we are trying to find a Lipschitz constant that holds for **any** choice of $x, y \in \Omega$. As the next step, we can use the Lipschitz constants of μ_* and v_*, to see that

$$\left|a(x) - a(y)\right| \leq \left(\sup_{\mu \in \mathbb{R}, v \in [0,\infty)} \left| \frac{\partial a}{\partial \mu}(\mu, v) \right| \alpha_{\mu_*} + \sup_{\mu \in \mathbb{R}, v \in [0,\infty)} \left| \frac{\partial a}{\partial v}(\mu, v) \right| \alpha_{v_*} \right) \|x - y\|_2. \tag{10}$$

What remains is to find the partial derivatives of the modified probability of improvement and modified expected improvement with respect to μ_* and v_*, and find an upper bound independent of μ_* and v_* for each. Some straightforward calculations show that

$$\begin{aligned}
\left| \frac{\partial MPI_\epsilon}{\partial \mu_*}(\mu_*, v_*) \right| &= \frac{1}{\sqrt{v_* + \epsilon}} \phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}} \right) \leq \frac{1}{\sqrt{2\pi\epsilon}}, \\
\left| \frac{\partial MPI_\epsilon}{\partial v_*}(\mu_*, v_*) \right| &= \frac{|f^* - \mu_*|}{2(v_* + \epsilon)^{3/2}} \phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}} \right) \leq \frac{1}{\sqrt{8\pi\epsilon^2}} \exp\Big(-\frac{1}{2} \Big), \\
\left| \frac{\partial MEI_\epsilon}{\partial \mu_*}(\mu_*, v_*) \right| &= \left| \Phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}} \right) - f^* \phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}} \right) \right| \leq 1 + \frac{|f^*|}{\sqrt{2\pi}}, \text{ and} \\
\left| \frac{\partial MEI_\epsilon}{\partial v_*}(\mu_*, v_*) \right| &= \frac{1}{\sqrt{4(v_* + \epsilon)}} \phi\left(\frac{f^* - \mu_*}{\sqrt{v_* + \epsilon}} \right) \leq \frac{1}{\sqrt{8\pi\epsilon}},
\end{aligned}$$

where we have used the bounds $\Phi(x) \leq 1$ and $\phi(x) = \frac{\exp\left(-x^2/2\right)}{\sqrt{2\pi}} \leq \frac{1}{\sqrt{2\pi}}$. Furthermore, the equality

$$\frac{\mathrm{d}\phi}{\mathrm{d}x}(x) = \frac{\mathrm{d}}{\mathrm{d}x}\left[\frac{1}{2\pi} \exp\Big(-\frac{x^2}{2} \Big) \right] = -x\phi(x)$$

was used to calculate the partial derivatives of the modified expected improvement, and the fact that

$$\sup_{x \in \mathbb{R}} \left| x \exp\Big(-\frac{x^2}{2b} \Big) \right| = \sqrt{b} \exp\Big(-\frac{1}{2} \Big)$$

was used to bound the derivative of the modified probability of improvement with respect to v_*. Note that the appearance of ϵ in three of these bounds reflects the fact that the normal probability of improvement and expected improvement acquisition functions are not Lipschitz-continuous.

Now that we have derived these bounds on the partial derivatives, we can apply Equation (10) directly to see that

$$\begin{aligned}
\left| \mathrm{MPI}_\epsilon(x) - \mathrm{MPI}_\epsilon(y) \right| &\leq \frac{1}{\sqrt{2\pi}} \left(\sqrt{\frac{n}{\epsilon}} \left\| K_{1:n}^{-1} f_{1:n} \right\|_2 + \frac{n}{2\epsilon} \frac{e^{-\frac{1}{2}}}{\lambda_{\min}(K_{1:n})} \right) \alpha_K \|x - y\|_2, \text{ and} \\
\left| \mathrm{MEI}_\epsilon(x) - \mathrm{MEI}_\epsilon(y) \right| &\leq \left(\sqrt{n} \left(1 + \frac{|f^*|}{\sqrt{2\pi}} \right) \left\| K_{1:n}^{-1} f_{1:n} \right\|_2 + \frac{n}{\sqrt{2\pi\epsilon}} \frac{1}{\lambda_{\min}(K_{1:n})} \right) \alpha_K \|x - y\|_2.
\end{aligned}$$

□

4.3. Global optimisation of Lipschitz-continuous functions by branch-and-bound

In the previous subsection, we have derived Lipschitz constants for some modified acquisition functions. Now we will briefly sketch an algorithm, based on [Fowkes et al., 2013], that uses these constants to get arbitrarily close to the global maximum of the function. Let a be an arbitrary acquisition function, and denote its Lipschitz constant by α_a. We are interested in optimising a over some compact domain $\Omega \subset \mathbb{R}^d$. We start by constructing a covering of Ω consisting of 3^d spheres of radius r_0, according to a splitting rule illustrated in [Fowkes et al., 2013]. All these spheres are saved in a first-in-first-out queue $\mathcal{Q}$ containing the spheres that still have to be evaluated, and we evaluate the acquisition function at the centre of each sphere and keep track of a^*, the highest value of the acquisition function encountered during optimisation. At this point, we are ready to start the iterative optimisation process until some stopping criterion has been satisfied, which could for example be a threshold value for the acquisition function or a minimal radius for the spheres.
Then we start the iterative procedure. Remove the first sphere from $\mathcal{Q}$, and denote its midpoint and radius by $x_S \in \mathbb{R}^d$ and $r_S > 0$. Then evaluate the condition $a(x_S) + r_S * \alpha_a > a^*$. If this is true, it is possible that the global optimum of the acquisition function can be found within this sphere. Apply the splitting rule to cover this sphere in 3^d spheres of radius $r_S/2$, discard any that lie completely outside Ω, evaluate a at the midpoints of the remaining spheres, and append them to $\mathcal{Q}$. If the condition does not hold, there is no improvement to be found here; continue with the next first sphere in $\mathcal{Q}$. For a more detailed description of this algorithm, readers are referred to [Fowkes et al., 2013].

4.4. Future research

In this section, we have introduced an approach for the global optimisation of acquisition functions in a setting of Bayesian Optimisation of univariate functions. In Section 3 we studied Bayesian Optimisation of multivariate functions. The acquisition functions derived for this multivariate setting are based on acquisition functions for the univariate settings, and might also be amenable to global optimisation using Lipschitz-continuity. This is left as an option for future research. Additionally, in the previous subsection we have roughly sketched the algorithm for the global optimisation. Working this out in more detail and implementing it, to see if or when the benefits of globally optimising the acquisition functions outweigh the costs, are also options left open for future research.

5. Algorithm

This section gives a high-level overview of the proposed optimisation algorithm. To recap, our goal is to find a minimum

$$\min_{x\in\Omega} \|f(x)\|_\infty,$$

where $\|\cdot\|_\infty$ denotes the max-abs norm, $\Omega \subset \mathbb{R}^d$ is the compact domain we search for the minimum and $f : \mathbb{R}^d \to \mathbb{R}^m$ is a vector-valued function.
In the algorithm, we initialise the set $\mathcal{D}$ of points at which the function is measured by using Latin Hypercube Sampling to generate $d + 1$ well-distributed points in Ω. We then evaluate f at each of these points to initialise the set of outputs $f(\mathcal{D})$ and calculate the current best objective value $d^* := \min_{x\in\mathcal{D}} \|f(x)\|_\infty$.
Then, we use a Gaussian Process to predict what the function looks like on the rest of Ω. We model the known function values $f := f(\mathcal{D})$ and the function value f_{n+1} at a new point x_{n+1} together as a multidimensional normal random variable:

$$\begin{pmatrix} f \\ f_{n+1} \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \mu_0 \\ \mu_1 \end{pmatrix}, \begin{pmatrix} \Sigma_{00} & \Sigma_{01} \\ \Sigma_{10} & \Sigma_{11} \end{pmatrix}\right).$$

Here μ_0 is the mean function prior (a function chosen in advance) evaluated on the points in $\mathcal{D}$, μ_1 is that same prior evaluated on x_{n+1}, and the Σ's are the evaluations of the kernel function. This kernel function takes as its input two points. The kernel Σ_{00} is a matrix containing the kernel function evaluated on all possible pairs in $\mathcal{D}$, Σ_{01} is the kernel function evaluated on all pairs of a point $\mathcal{D}$ and x_{n+1}, such that each entry in the column vector corresponds to one point in $\mathcal{D}$, Σ_{10} is the transpose of Σ_{01} and Σ_{11} is the kernel function evaluated on (x_{n+1}, x_{n+1}).
If we would know nothing about the function f, modelling it as a random variable would make some amount of sense, but the choice to model it with the mean function μ and kernel function Σ would also be completely arbitrary and therefore useless. However, the situation is better, since we know the outputs $f = f(\mathcal{D})$. By conditioning on these values, we can make an informed prediction for the new function values

$$\begin{aligned} f_*|f = f(\mathcal{D}')|f(\mathcal{D}) &\sim \mathcal{N}\Big(\mu_*, \Sigma_*\Big) \\ \mu_* &:= \Sigma_{10}\Sigma_{00}^{-1}\Big(f - \mu_0\Big) + \mu_1 \\ \Sigma_* &:= \Sigma_{11} - \Sigma_{10}\Sigma_{00}^{-1}\Sigma_{01}. \end{aligned} \tag{11}$$

For a detailed derivation, see Subsection 2.1. The so-called priors (mean and kernel function) remain to be chosen. For the mean function, we make the conventional choice $\mu(x) := 0$, which is appropriate if the function to be optimised is a black box (at the start of the optimisation problem). A common choice for the kernel function is the radial basis function

$$K_{RBF}(\sigma, l; x, x') := \sigma \exp\Big(\frac{\|x - x'\|_2}{2\ell}\Big),$$

where the variance-hyperparameter σ linearly scales the posterior variance and the lengthscale-hyperparameter ℓ influences the rate at which the correlation between function values decreases: if ℓ is large, the model expects that a known function value at a great distance still gives some indication of what the new function value could be. A benefit of using this kernel function is that the resulting matrix Σ_{00} will be positive definite. A disadvantage, however, is that this matrix will be full of non-zero entries. Thus, while it is invertible, this will take a lot of time. To circumvent this problem, we approximate the radial basis function by the Wendland function

$$K\left(\sigma, q := \frac{\|x - x'\|_2}{\ell}\right) := \begin{cases} \sigma\left[1 - \frac{3}{2}q^2\left(1 - \frac{q}{2}\right)\right] & \text{if } 0 \leq q \leq 1, \\ \frac{\sigma}{4}(2-q)^3 & \text{if } 1 < q \leq 2, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

where the hyperparameters σ and ℓ have the same function as in the radial basis function. By choosing an appropriate value of ℓ, we can control the sparsity of the matrix.

At the start of the optimisation process, we first optimise the hyperparameters to fit the known data $(x, f) := (\mathcal{D}, f(\mathcal{D}))$. We do this by maximising the log likelihood, which is the logarithm of the likelihood of seeing this data under the choice of hyperparameters:

$$\log p((x, f)|\sigma, \ell) = -\frac{1}{2} f^\top \Sigma_{00}^{-1}(\sigma, \ell) f - \frac{1}{2} \log |\Sigma_{00}(\sigma, \ell)| - \frac{|\mathcal{D}|}{2} \log 2\pi,$$

where $|\mathcal{D}|$ is the number of points in $\mathcal{D}$. This expression results directly from the assumption that the function values at the points in $\mathcal{D}$ are normally distributed with mean 0 and covariance $\Sigma_{00}(\sigma, \ell)$. By maximising this expression with respect to the hyperparameters, we find the variance and lengthscale for which the model of Gaussian Process Regression yields the maximum possible probability of encountering the known data. That is, if we would draw random samples from the distribution generated by the Gaussian Process for each of the points in $\mathcal{D}$, then the probability of seeing the actual function values $f(\mathcal{D})$ is maximised by using this variance and lengthscale. There is one problem: we have to impose bounds on these hyperparameters. The variance can only be positive. To ensure this, we use the logarithm of the variance, and only exponentiate it at the moment we need to calculate the kernel matrix. The lengthscale should also always be positive and should not become larger than the largest possible distance in Ω. Let us denote this distance by $d_{\max}$ and the lengthscale hyperparameter outside kernel calculation by ζ, then we use the following conversion for the lengthscale when we want to calculate the kernel matrix:

$$\ell := (1 + \tanh \zeta) * \frac{d_{\max}}{2}.$$

The hyperbolic tangent is a smooth function with domain $\mathbb{R}$ and range $(-1, 1)$, so by applying this transformation the length scale will always be in $(0, d_{\max})$.

When the optimal hyperparameters have been found, we can start the main loop, in which we use (11) to predict new function values μ_* and uncertainties Σ_*, use these

predictions to optimise an acquisition function $q(\mu_*, \Sigma_*)$, add the point at which the acquisition function is optimal to $\mathcal{D}$ and the corresponding function value to $f(\mathcal{D})$, and update the best found norm d^*. Then the loop repeats until d^* becomes small enough (smaller than some performance threshhold ϵ) or until the maximum number of iterations has been reached.
To recap, here is the algorithm in pseudocode:

Algorithm 1 The optimisation algorithm

Initialise $\mathcal{D}$ by Latin Hypercube Sampling
Calculate $f(\mathcal{D})$
$d^* \leftarrow \min_{x \in \mathcal{D}} \|f(x)\|_\infty$
$\sigma, l \leftarrow \text{argmax}_{\sigma,l}(-\frac{1}{2} f^\top K_{00}^{-1}(\sigma, l) f - \frac{1}{2} \log |K| - \frac{n}{2} \log 2\pi)$
while num iters < max num iters and $d^* > \epsilon$ **do**
    Calculate $\Sigma_{00} = K(\mathcal{D}, \mathcal{D})$, $\Sigma_{01} = K(\mathcal{D}, x_{n+1})$ and $\Sigma_{11} = K(x_{n+1}, x_{n+1})$
    $\mu_* \leftarrow \Sigma_{01}^\top \Sigma_{00}^{-1} f(\mathcal{D})$
    $\Sigma_* \leftarrow \Sigma_{11} - \Sigma_{01}^\top \Sigma_{00}^{-1} \Sigma_{01}$
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \text{argmax}\, q$
    $f(\mathcal{D}) \leftarrow f(\mathcal{D}) \bigcup f(\text{argmax}\, q)$
    $d^* = \min(d^*, \|f(\text{argmax}\, q)\|_\infty)$
end while

5.1. Inversion trick

we can employ a trick to speed up the inversion of the kernel matrix. During each iteration step $k \geq 2$ of the optimisation process, we have to invert the kernel matrix Σ_k on the known data. In terms of the kernel matrix Σ_{k-1} during the previous step, this matrix looks like

$$\Sigma_k = \begin{pmatrix} \Sigma_{k-1} & b_k \\ b_k^\top & \sigma \end{pmatrix},$$

where b_k denotes a column vector containing the kernel function evaluated on the distance between all points from the previous step and the new point and where σ is the variance parameter.[2]
If we save the inverse of the previous kernel matrix Σ_{k-1}, we can calculate the inverse of Σ_k without doing any additional inversion. To do this, we exploit the known expression for the inverse of a 2x2 block matrix given by [Lu and Shiou, 2002]. First, we calculate the Schur complement of Σ_{k-1}:

$$S_k := \Sigma_k / \Sigma_{k-1} = \sigma - b_k^\top \Sigma_{k-1}^{-1} b_k.$$

[2]The entries on the diagonal correspond to the zero distance between a point and itself, in which case our kernel function returns the variance.

Note that in our case, this is a real number between 0 and σ. The inverse of Σ_k is then given by

$$\Sigma_k^{-1} = \frac{1}{S_k} \begin{pmatrix} S_k \Sigma_{k-1}^{-1} + \Sigma_{k-1}^{-1} b_k b_k^\top \Sigma_{k-1}^{-1} & -\Sigma_{k-1}^{-1} b_k \\ -b_k^\top \Sigma_{k-1}^{-1} & 1 \end{pmatrix}.$$

Including this trick and evaluating its effectiveness is outside the scope of this thesis, and is left as an opportunity for future research.

6. Experiments

In this section, we explain the experiments on which we compare our method to other optimisation methods. First, we illustrate some preliminary results for a small toy problem. The rest of the section explains the setup for larger experiments. To do this, we use a synthetic test problem based on the Salomon function, and we also compare performance on optimising the application BendAid. In the case of the synthetic problem, we compare methods in three cases: a base case in which both the input and output dimension are 3, a high-input dimension case in which the input dimension is 10 and the output dimension is 3 and a high-output dimension case in which the input dimension is 3 and the output dimension is 10. For BendAid, the number of input parameters is 3 and the output dimension is at most 2000. We reduce the dimension of this output to 3 and 10 for two test cases. All comparisons are made both in the number of necessary function evaluations and processing time, and we use the probability of improvement acquisition function with a few choices for the exploration-exploitation trade-off parameter: 0.5 for favouring exploration, 1 for balance and 2 for favouring exploitation. Some of the results will be highlighted in the next section.

6.1. Toy problem

Our goal is to find $x \in [0, 2\pi]$ to get the vector-valued function

$$h(x) = \begin{pmatrix} 5 + \sin(x) \\ 2 + 1.3\cos(x) \\ \tanh(x) \end{pmatrix}$$

as close as possible to the target $y = (4.5, 2, 0.8)^\top$, where distance is measured by the $\|\cdot\|_\infty$-norm. The standard approach is to apply Gaussian Process Regression to the composite function $\|h(\cdot) - y\|_\infty : [0, 2\pi] \to [0, \infty)$ directly and use expected improvement as the acquisition function to decide where to sample next. The two approaches we propose both perform Gaussian Process Regression on each component $|h_i(\cdot) - y_i|$ independently. Then the first approach uses the distribution derived in section 3.2 as the probability of improvement, and the second approach approximates the integral derived in subsection 3.3 as a replacement for the expected improvement.

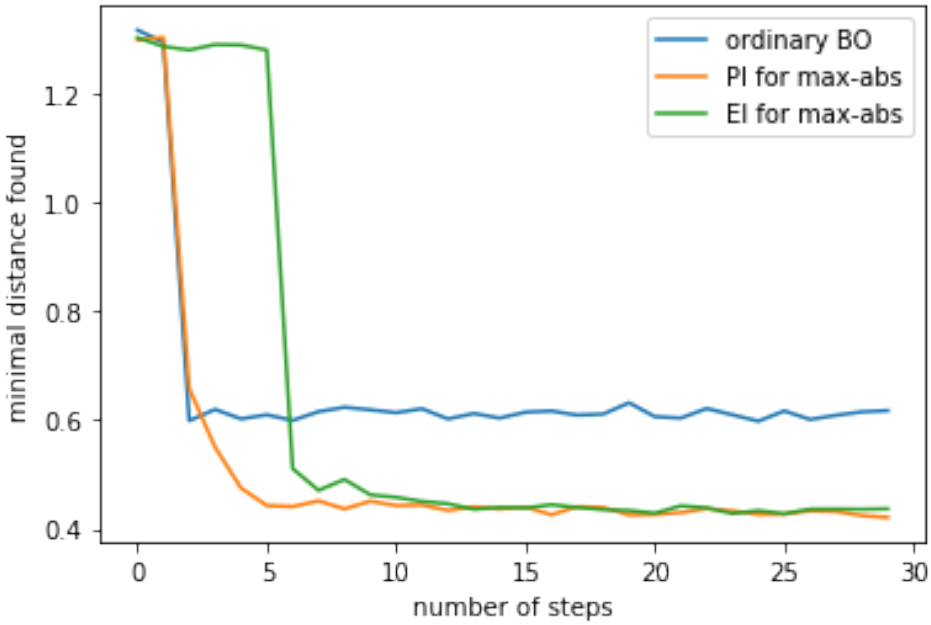

Figure 6.1: A comparison of methods to find the minimum of $\|h(x) - y\|_\infty$.

In figure 6.1 we see the minimal distance found by these three methods as a function of the number of steps, which is equal to the number of function measurements. All methods start with one measurement at $x = \pi$. We see that for this simple test case, ordinary Bayesian Optimisation (BO) and Bayesian Optimisation with the probability of improvement for the maximum absolute-value norm acquisition function (PI for max-abs; see Subsection 3.2) find the same decrease in the first step, but after that, ordinary BO is unable to find a point where the function is even closer to the target (for the parameters we have chosen), whereas PI for max-abs causes a further decrease. Bayesian Optimisation with the expected improvement for the maximum absolute-value norm acquisition function (EI for max-abs) needs some steps to explore first, but then it finds the same minimum which PI for max-abs finds. Ordinary BO is unable to find this minimum in 30 steps.

6.2. Synthetic test problem

The synthetic benchmark problem has scalable input and output dimensions. We construct a function

$$f : \mathbb{R}^m \to \mathbb{R}^n,$$

which will allow us to do tests for various combinations of the input dimension m and output dimension n. This function is based on the Salomon function [NDT, nd]

$$f_S(\boldsymbol{x}) := 1 - \cos\left(2\pi \sqrt{\sum_{i=1}^{m} x_i^2}\right) + 0.1 \sum_{i=1}^{m} x_i^2,$$

which typically has the range $[-100, 100]$ for each input variable. We construct our multi-output function by changing the coefficient 0.1, so our test function has components given by

$$f_i(\boldsymbol{x}) := 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{m} x_i^2}\right) + 10^{\alpha_i}\sum_{i=1}^{m} x_i^2,$$

where $\alpha_i := i - \lceil \frac{n}{2} \rceil$. This creates components where the last term is very large compared to the '$1 - \cos(\ldots)$' part (which is bounded by 0 and 2), as well as components where it is very small in comparison. We also change the input range to $[-100, 80]$ for each variable in order to introduce asymmetry into the problem. The goal of this synthetic problem is to minimise

$$\max_{1\leq i\leq n} \left\| f(\boldsymbol{x}) \right\|_{\infty},$$

the maximum absolute value over all the components of f.

There are three reasons why this benchmark problem is interesting. Firstly, we can easily change the combination of input and output dimension in order to do different tests. Secondly, the known minimum of the max-norm is 0 at $\boldsymbol{x} = \boldsymbol{0}$. Thirdly, the components of the function are not convex, which makes it a more interesting problem for Bayesian Optimisation, since this method can escape from local minima.

6.3. BendAid

In this subsection, the application BendAid is introduced. BendAid is a software library developed to help Dynobend, a company that specialises in bending steel tubes into prescribed forms. However, correctly predicting the form that will result from certain settings is not trivial. BendAid aims to solve this problem of sometimes not correctly producing a target shape S_{target} by finding the set of parameters P that transform the real shape into the target shape. The meaning of these parameters and how they define the transformation will be discussed in the next subsection.

After the tube has been bent, the position of a number of points along it are measured, and the position of a number of points on the target shape are sampled numerically. The Iterative Closest Point algorithm is then applied to rotate and translate the real points in a way that minimises the sum of squared Euclidean distances between the real points and the target points. The Euclidean distances are the information used by our method, which chooses a set of parameters P to transform the model of the real shape.

At this point, the optimisation loop starts. the transformation implied by P is first applied to the model of the real shape. Then the Iterative Closest Point algorithm is applied, and residual vectors between every pair of points are returned. Our model chooses the next set of parameters based on the resulting Euclidean distances. This continues until time runs out or until the sum of Euclidean distances falls below a certain performance threshold.

Finally, the inverse of the best found transformation is applied to the model the bending machine receives as input, in the hope that this will counter the error in the physical machine.
In the rest of this subsection, we will elaborate on the explanation above. We begin by discussing the theory of three-dimensional curves based on the Frenet-Serret formulas. Then we take a closer look at how BendAid uses this theory and what parameters we have to optimise as a result, and finally we introduce the choices made to model the problem.

6.3.1. Theory of 3D curves

When you want to characterise a curve in three-dimensional space, the Frenet-Serret equations are a useful first step. To derive these equations, we start by appointing an orthonormal basis of $\mathbb{R}^3$ to every point along the curve.

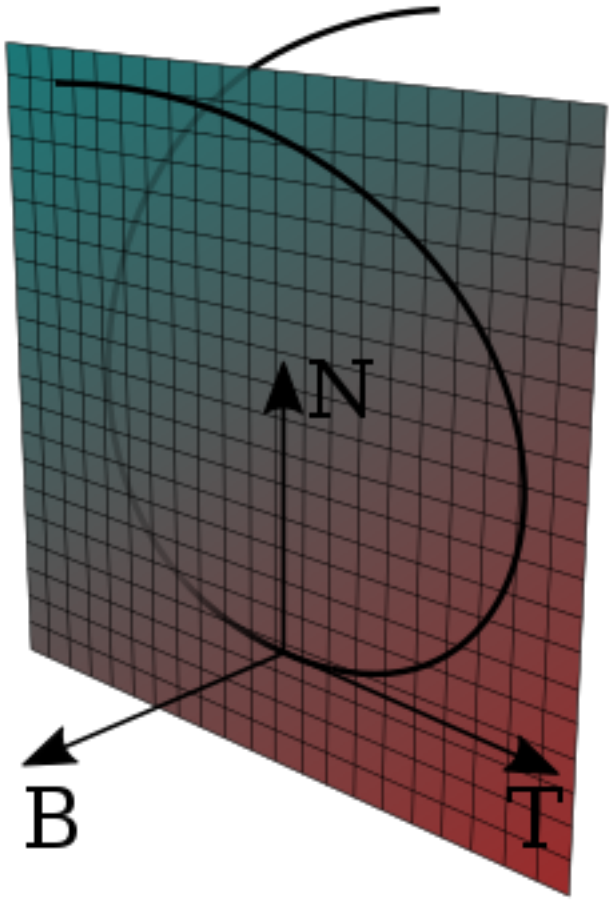

Figure 6.2: The Frenet-Serret T(angent), N(ormal) and B(inormal) unit vectors at a point on a curve. source: Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Frenet.png).

The first basis vector $T(s)$ is the unit tangent along the curve. Here s is the arclength parameter which denotes our position along the curve. For the second basis vector, we choose the normal unit vector $N(s)$, defined as

$$N(s) := \frac{T'(s)}{\|T'(s)\|_2},$$

where $T'(s) = \frac{dT}{ds}(s)$ denotes the derivative of T with respect to the arclength parameter and $\|\cdot\|_2$ denotes the Euclidean norm. Note that this definition does not make sense if $T'(s)$ is the zero vector; indeed, in this case $T(s)$ is constant, so the curve is a straight line! In what follows we make the assumption that this is the case at no point s along the curve. The vector $N(s)$ has length 1 by definition, and to see that it is perpendicular to the tangent, we can write

$$\begin{aligned} N(s) \cdot T(s) &= \frac{1}{\|T'(s)\|_2} T'(s) \cdot T(s) \\ &= \frac{1}{2\|T'(s)\|_2} \frac{d}{ds}\Big(T(s) \cdot T(s)\Big) \\ &= \frac{1}{2\|T'(s)\|_2} \frac{d}{ds}\Big(1\Big) = 0 \end{aligned}$$

where $\cdot$ denotes the inner product. The last basis vector is the binormal unit $B(s)$, defined as

$$B(s) := T(s) \times N(s).$$

This vector has length 1 and is perpendicular to T and N by properties of the outer product, which means we now have the orthonormal basis $\{T(s), N(s), B(s)\}$ at every point s along the curve.
When we have these basis vectors, we can express their derivatives as linear combinations of them. At every point along the curve, indexed by s, $\{T(s), N(s), B(s)\}$ is an orthonormal basis in which we can express $T'(s)$, $N'(s)$ and $B'(s)$. To start we have $T'(s) = \|T'(s)\|_2 N(s)$ by definition of $N(s)$. We will denote the **curvature** by

$$\kappa(s) := \|T'(s)\|_2.$$

This gives us the first Frenet-Serret equation:

$$T'(s) = \kappa(s) N(s).$$

Next, let's take a look at $N'(s)$. We saw earlier that $T'(s)$ is perpendicular to $T(s)$. In fact, this argument also holds for $N'(s)$ and $N(s)$. Since $\{T(s), N(s), B(s)\}$ is a basis, we can therefore write

$$N'(s) = \alpha(s) T(s) + \tau(s) B(s),$$

for certain variables $\alpha(s)$ and $\tau(s)$. It turns out that $\alpha(s)$ can be determined. To do this, we first note that

$$\alpha(s) = N'(s) \cdot T(s).$$

Secondly, we have

$$0 = \frac{d}{ds}\Big(T(s) \cdot N(s)\Big) \implies T'(s) \cdot N(s) = -T(s) \cdot N'(s).$$

Putting these two equations together, we find that

$$\alpha(s) = -T'(s) \cdot N(s) = -\kappa(s).$$

We now have the second Frenet-Serret equation:

$$N'(s) = -\kappa(s)T(s) + \tau(s)B(s).$$

To derive the third, we use the product rule:

$$\begin{aligned} B'(s) &= T'(s) \times N(s) + T(s) \times N'(s) \\ &= \kappa(s)N(s) \times N(s) + T(s) \times \Big(- \kappa(s)T(s) + \tau(s)B(s)\Big) \\ &= \tau(s)T(s) \times B(s), \end{aligned}$$

where in the third equality we used the fact that the cross product of a vector with itself yields the zero vector. So what is $T(s) \times B(s)$? Well, it's a vector orthogonal to $T(s)$ and $B(s)$ of length 1, so it has to be either $N(s)$ or $-N(s)$. It can be verified by taking the derivative of $0 = B(s) \cdot N(s)$ that it is the negative case, so we have found the third Frenet-Serret equation:

$$B'(s) = -\tau(s)N(s).$$

Together, these three equations can be summarised using a skew-symmetric matrix:

$$\begin{pmatrix} T'(s) \\ N'(s) \\ B'(s) \end{pmatrix} = \begin{pmatrix} 0 & \kappa(s) & 0 \\ -\kappa(s) & 0 & \tau(s) \\ 0 & -\tau(s) & 0 \end{pmatrix} \begin{pmatrix} T(s) \\ N(s) \\ B(s) \end{pmatrix}. \tag{12}$$

The observant reader will have noticed that while we have given $\kappa(s)$ the name curvature, we have not yet given $\tau(s)$ a name. Before we do this, we may notice that

$$\tau(s) = \| - \tau(s)N(s)\|_2 = \|B'(s)'\|_2.$$

$\tau(s)$ is the size of the derivative of the binormal unit vector, and we call it the **torsion**.

6.3.2. Experiment choices

Now that we have this theory, we can use it to bend the steel tubes into the right shape. In this process, the parameters we are optimising over are the curvature $\kappa(s)$ and the torsion $\tau(s)$. In general, these parameters vary over the length of the curve. To reduce the dimension of the input space, BendAid fixes the value of these parameters at certain points along the curve and uses linear interpolation in between said points. In this way, the profile is specified as (c, t). Here c specifies the number of fixed points for the curvature and t specifies the number of fixed points for the torsion. In this thesis, the model we will study has the profile $(3, 2)$.

7. Results

In this section, we share the results of the experiments explained in the previous section. The results are summarised in tables which give a high-level overview of the compared performance of different methods. For all comparisons, we first let vanilla Bayesian Optimisation (with trade-off parameter $\xi = 1$) run for a given number of iterations or seconds of process time. In the case of process time, we only consider the time required for the methods to choose the evaluation points, not the time required to evaluate the target function at that point. This experimental choice is made because we are interested in the performance of the algorithm when the costs of evaluating the target function are negligible compared to the costs of the algorithm. We note the lowest score ($\|\cdot\|_\infty$-value) this method achieves in the given number of iterations or, in the case of process time, the number of iterations at which it first exceeds the barrier. Then, we calculate the number of iterations/the process time required to attain a lower score for three versions of our method with different trade-off parameter values ξ. Here, one iteration is formally defined as first selecting a new point and then evaluating the target function at that point. In this chapter, VBO refers to Vanilla Bayesian Optimisation with the probability of improvement acquisition function being directly applied to the score, whereas BO refers to Bayesian Optimisation with the probability of improvement function we derived for the maximum absolute value norm of a vector-valued function. To measure process time, we use Julia's time function to keep track of the cumulative time the optimisation model requires. We ran into some numerical problems during our investigations. This happened either when optimising a Gaussian Process with the GaussianProcesses.jl package or when optimising the acquisition function using Optim.jl. In the former, we used an un-optimised Gaussian Process for that iteration, and in the latter, instead of selecting a promising point to next evaluate the target function at, we chose a random point. More information on the implementation in Julia can be found in the appendix.

7.1. Results on the simulation

Tables 7.1 and 7.2 contain the results for the synthetic optimisation problems. Table 7.1 contains the low-input, high-output scenario with input dimension 3 and output dimension 10. Table 7.2 contains the high-input, low-output scenario with input dimension 10 and output dimension 3. The first row contains the scores achieved by vanilla BO. All other rows contain the time it takes other methods to surpass said scores. The first column lists the optimisation method. ξ denotes the trade-off parameter used in Bayesian Optimisation. Columns 2-4 contain the lowest score ($\|\cdot\|_\infty$-value) found after 5, 20 and 50 iterations respectively. Columns 5-7 contain the process time it takes the methods to exceed the value vanilla Bayesian Optimisation (with $\xi = 1$) finds after 5, 20 and 50 ms of process time respectively. BO uses the probability of improvement formula derived in section 3.2. These times were chosen because it takes VBO roughly 1 ms to perform 1 iteration, so the results VBO finds will more or

less align.

VBO ($\xi = 1$) score	569656	402890	150130	586946	172140	0
VBO ($\xi = 1$)	5 cycles	10 cycles	20 cycles	1 ms	5 ms	25 ms
BO ($\xi = 0.5$)	4 cycles	8 cycles	17 cycles	89 ms	147 ms	163 ms
BO($\xi = 1$)	6 cycles	7 cycles	28 cycles	28 ms	163 ms	186 ms
BO ($\xi = 2$)	8 cycles	10 cycles	54 cycles	45 ms	117 ms	217 ms

Table 7.1: The results on the synthetic test problem with low input dimension (3) and high output dimension (10). VBO refers to vanilla Bayesian optimisation. NM refers to Nelder-Mead.

By comparison, Nelder-Mead finds a value of 1.99 in virtually no time, in 1071 function calls.

VBO ($\xi = 1$) score	143.5	143.5	126.6	174.0	135.6	94.0
VBO ($\xi = 1$)	5 cycles	20 cycles	50 cycles	5 ms	20 ms	50 ms
BO ($\xi = 0.5$)	5 cycles	6 cycles	9 cycles	1 ms	2 ms	624 ms
BO($\xi = 1$)	16 cycles	17 cycles	18 cycles	0 ms	1 ms	32 ms
BO ($\xi = 2$)	2 cycles	3 cycles	9 cycles	1 ms	12 ms	2890 ms

Table 7.2: The results on the synthetic test problem with high input dimension (10) and low output dimension (3). VBO refers to vanilla Bayesian optimisation.

By comparison, Nelder-Mead finds a value of virtually 0 in virtually no time, using 386 function calls.

7.2. Results on BendAid

Table 7.3 contains the results for the BendAid optimisation problem. For BendAid, we evaluate the method on a $(2, 2, s)$-model with 5 input parameters. The output is restricted to be ten-dimensional. These choices were made to not make the problem trivial, but also to keep it computationally feasible for a regular laptop. More on this can be found in section 6.3.2.

VBO ($\xi = 1$) score	133.3	127.6	120.6	143.7	143.7	141.2
VBO ($\xi = 1$)	5 cycles	20 cycles	50 cycles	5 ms	20 ms	50 ms
BO ($\xi = 0.5$)	5 cycles	14 cycles	$\gg 50$ cycles[3]	107 ms	121 ms	140 ms
BO ($\xi = 1$)	7 cycles	8 cycles	12 cycles	30 ms	34 ms	40 ms
BO ($\xi = 2$)	20 cycles	21 cycles	42 cycles	67 ms	79 ms	91 ms

Table 7.3: The results on the BendAid optimisation problem. The model is $(3, 2)$, so the input dimension is 5, and the output dimension is restricted to 10. VBO refers to vanilla Bayesian optimisation.

By comparison, Nelder-Mead finds a value of 104.7, but requires 53 function calls for this.

[3] In this setup, the method got stuck at a score of ≈ 122 for a comparatively long time.

8. Conclusion

In conclusion, the results in the previous chapter suggest most strongly that Nelder-Mead is a faster optimisation method, both in practical applications and synthetic problems. However, it does use more function calls. As expected, when the output dimension is high, the methods that create an individual Gaussian Process for each output dimension require a lot more time, but less iterations than the method that models only the absolute maximum value norm. Comparing these former methods amongst themselves, we see that the one that prefers exploitation ($\xi = 2$) is more costly (both in cycles and process time) in later stages, but is very effective early on in the case of low output dimension. The BendAid application shows us the opposite: that the exploration favouring method ($\xi = 0.5$) becomes a lot more expensive at the end.

9. Discussion

Our results are twofold. First, we present methods that extend Bayesian Optimisation to the optimisation of multivariate functions with respect to a maximum absolute value norm and compare this numerically to regular Bayesian Optimisation. The approach presented in this thesis, to optimise functions by applying Bayesian Optimisation with a probability of improvement function derived for the maximum absolute value of a vector-valued function, appears to become more promising for problems with an expensive to evaluate target function. Based on the synthetic data, an approach that looks promising is starting with a method that prefers exploitation ($\xi = 2$), then switching to a method that favours exploration ($\xi < 1$) later on, or vice versa, depending on the problem at hand. However, as the data supporting these findings is limited and we ran into numerical problems, more research is needed to investigate the validity of these conclusions. Due to time constraints, we were unable to consider optimisation with the expected improvement function and to apply the inversion trick for covariance matrices described in subsection 5.1. Another problem with acquisition functions is that they are generally non-convex, which makes them hard to optimise in themselves. A way to partly overcome this is applying multiple restarts, but a method that uses some inherent structure of the acquisition function is more promising.
Finally, in Subsection 5.1 we propose a method to update the posterior covariance matrix during Bayesian Optimisation, which normally requires a matrix inversion during every iteration, without matrix inversion. This method comes with the cost of not being able to change certain hyperparameters between iterations of the optimisation process. Exploring the effectiveness of this method is left as another avenue for future research.

References

[NDT, nd] (n.d.). N-d test functions s — ampgo 0.1.0 documentation. `https://infinity77.net/global_optimization/test_functions_nd_S.html`. (Accessed on 09/13/2022).

[Astudillo and Frazier, 2019] Astudillo, R. and Frazier, P. (2019). Bayesian optimization of composite functions. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 354–363. PMLR.

[Bates et al., 2004] Bates, S., Sienz, J., and Toropov, V. (2004). Formulation of the optimal latin hypercube design of experiments using a permutation genetic algorithm. *Accessed through ResearchGate*, 2011.

[Burdick and Lesley, 1975] Burdick, D. and Lesley, F. D. (1975). Some uniqueness theorems for analytic functions. *The American Mathematical Monthly*, 82(2):152–155.

[Fowkes et al., 2013] Fowkes, J., Gould, N., and Farmer, C. (2013). A branch and bound algorithm for the global optimization of Hessian Lipschitz continuous functions. *Journal of Global Optimization*, 56:1793–1797.

[Frazier, 2018] Frazier, P. I. (2018). A tutorial on Bayesian optimization. *digital preprint, arXiv #1807.02811*.

[Lu and Shiou, 2002] Lu, T.-T. and Shiou, S.-H. (2002). Inverses of 2×2 block matrices. *Computers & Mathematics with Applications*, 43(1):119–129.

[Qin et al., 2017] Qin, C., Klabjan, D., and Russo, D. (2017). Improving the expected improvement algorithm. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5387–5397, Red Hook, NY, USA. Curran Associates Inc.

[Soch, 2020] Soch, J. (2020). Conditional distributions of the multivariate normal distribution — the book of statistical proofs. `https://statproofbook.github.io/P/mvn-cond`. (Accessed on 08/31/2023).

[Uhrenholt and Jensen, 2019] Uhrenholt, A. K. and Jensen, B. S. (2019). Efficient Bayesian optimization for target vector estimation. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2661–2670. PMLR.

[Wendland, 1995] Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396.

A. Appendix

A.1. Implementation in Julia

we are using Gaussian Process Regression to find

$$\min_{x\in\Omega} \left\| f(x) \right\|_\infty ,$$

where $f : \mathbb{R}^n \to \mathbb{R}^k$ and $\Omega \subset \mathbb{R}^n$. We save the input points where the function has been measured in a matrix of dimensions $n \times m$, where m is the number of input points so far. For example, if the input dimension is 3 and we have measured at four points, the input point matrix could look like

$$\begin{pmatrix} -8.0 & 64.0 & -44.0 & 28.0 \\ 28.0 & -8.0 & -44.0 & 64.0 \\ -8.0 & 28.0 & 64.0 & -44.0 \end{pmatrix}.$$

We also save the function values in a matrix. This matrix has dimensions $k \times m$, so each row corresponds to one output dimension of the function and each column corresponds to one input point. For example, if $k = 2$ and the first function component has $\alpha = 0.1$ while the second has $\alpha = 1.0$, the function matrix, rounded to one decimal place and given the above input points, would be

$$\begin{pmatrix} 3.7 & 8.4 & 10.0 & 10.2 \\ 30.9 & 71.7 & 90.3 & 84.4 \end{pmatrix}.$$

These function values result from the multi-dimensional Salomon function given by

$$f_\alpha(\boldsymbol{x}) := 1 - \cos\left(2\pi\|\boldsymbol{x}\|_2\right) + 10^\alpha \|\boldsymbol{x}\|_2.$$

Now, we are going to do a sanity check by hand to see if we are getting the right results. We will be analysing if figure A.1 is correct.
In this case, we start with two measured points $x_1 := -20$ and $x_2 := 40$ and the corresponding function values $y_1 := f_{-1}(x_1) = 2$ and $y_2 := f_{-1}(x_2) = 4$, and used them to fit a Gaussian Process with the zero mean prior $\mu_0(x) := 0$ and the squared exponential kernel prior $K_0(x, x') := \sigma^2 \exp\Big(-\frac{(x-x')^2}{2\ell^2} \Big)$, where in this case we have chosen the hyperparameters to be $\sigma = e$ and $\ell = e^2$.

The first question to answer is whether the mean and variance predictions in figure A.1 agree with the theory. These predictions at a new point x are made through the Bayesian update rules

$$\mu(x) = K_0(x, x_{1,2}) K_0(x_{1,2}, x_{1,2})^{-1} (y_{1,2} - \mu_0(x_{1,2})) + \mu_0(x), \text{ and} \tag{13}$$

$$\sigma^2(x) = K_0(x, x) - K_0(x, x_{1,2}) K_0(x_{1,2}, x_{1,2})^{-1} K_0(x_{1,2}, x), \tag{14}$$

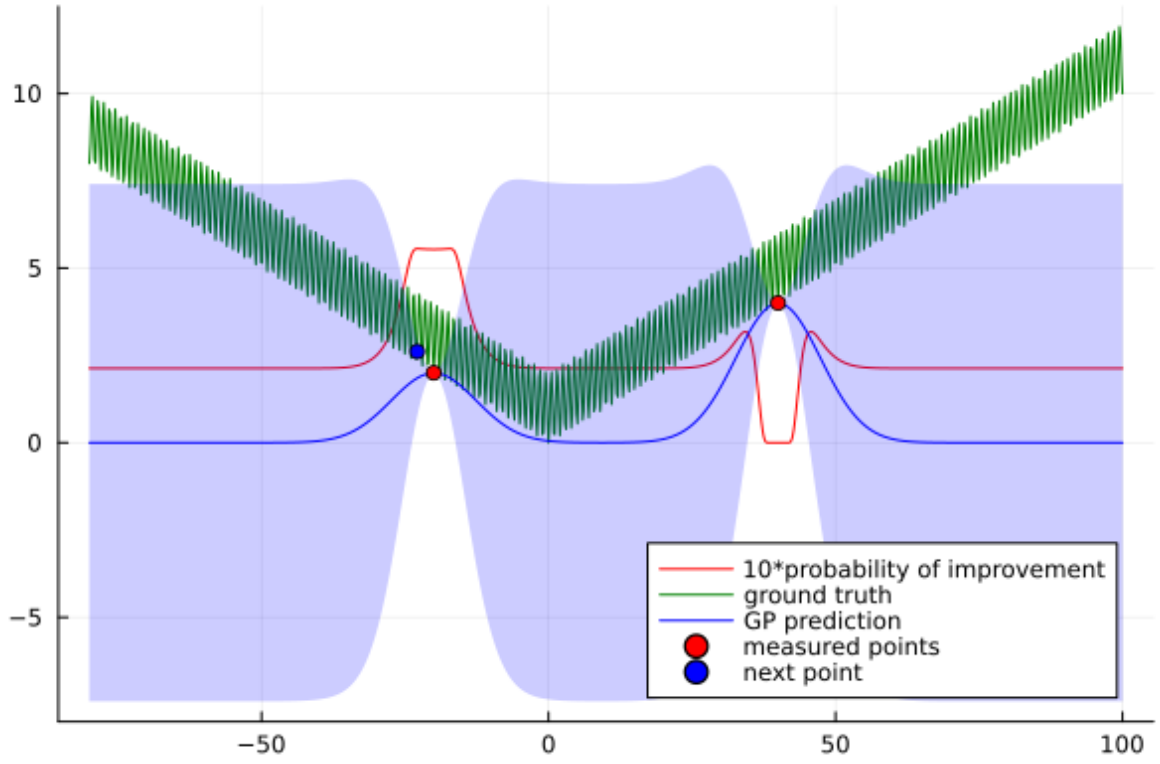

Figure A.1: An example of the prediction of a Gaussian Process fitted to the Salomon function in 1D, based on two points of measurement (red dots). The green line represents the 1D Salomon function we are optimising (with $\alpha = -1$), the blue line and shade are the mean and variance prediction, respectively, the red line represents the probability of improvement (scaled by a factor of 10) and the blue point is the next point to be measured, since it represents the highest probability of improvement.

where $x_{1,2}$ is shorthand for the column vector containing x_1 and x_2, likewise $y_{1,2}$ denotes the column vector containing y_1 and y_2 and μ_0 and K_0 are applied element-wise, where $K_0(\ldots,\ldots)$ takes the form of a matrix of shape $n \times m$, where n is equal to the length of the first argument and m is equal to the length of the second argument. We know the values of x_1, x_2, y_1 and y_2 and the priors $\mu_0(x)$ and $K_0(x, x')$; if we enter these into the update rules, we find

$$\mu(x) = \begin{pmatrix} \exp\left(2 - \frac{(x+20)^2}{2e^4}\right) & \exp\left(2 - \frac{(x-40)^2}{2e^4}\right) \end{pmatrix} \begin{pmatrix} e^2 & e^{2-1800e^{-4}} \\ e^{2-1800e^{-4}} & e^2 \end{pmatrix}^{-1} \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \text{ and}$$

$$\sigma^2(x) = 1 - \begin{pmatrix} \exp\left(2 - \frac{(x+20)^2}{2e^4}\right) & \exp\left(2 - \frac{(x-40)^2}{2e^4}\right) \end{pmatrix} \begin{pmatrix} e^2 & e^{2-1800e^{-4}} \\ e^{2-1800e^{-4}} & e^2 \end{pmatrix}^{-1} \begin{pmatrix} \exp\left(2 - \frac{(x+20)^2}{2e^4}\right) \\ \exp\left(2 - \frac{(x-40)^2}{2e^4}\right) \end{pmatrix},$$

and by acknowledging that $e^{-1800e^{-4}}$ is an extremely inefficient but accurate approxi-

mation of 0, we can approximate the inverse matrix as $e^{-2}\mathcal{I}_2$ to finally find that

$$\mu(x) = 2 \cdot \exp\Big(- \frac{(x+20)^2}{2e^4}\Big) + 4 \cdot \exp\Big(- \frac{(x-40)^2}{2e^4}\Big), \text{ and}$$
$$\sigma^2(x) = e^2\Big[1 - \exp\Big(- \frac{(x+20)^2}{e^4}\Big) - \exp\Big(- \frac{(x-40)^2}{e^4}\Big)\Big].$$

Indeed, this corresponds to what we see in figure A.1: The mean estimate has a Gaussian curve of height 2 around $x = -20$, a Gaussian curve of height 4 around $x = 40$ and is flat elsewhere, and the variance is approximately $e^2 \approx 7.4$ away from the known points, with a Gaussian valley around the known points. The small increases in height we see just before and after such a valley are due to the fact that the variance estimate is drawn symmetrically around the mean estimate.

A.2. Another proof of theorem 3.1

As a recap, we are interested in solving the optimisation problem

$$\min_{x\in\Omega} \left\|f(x)\right\|_\infty,$$

where $f : \mathbb{R}^d \to \mathbb{R}^m$ is an unknown function. Each component of f is modelled as a Gaussian Process Regression, and then regression is applied based on the observations $\mathcal{D} = \{x_1, \dots, x_n, (f_1(x_1), \dots, f_m(x_1)), \dots, (f_1(x_n), \dots, f_m(x_n))\}$. According to the model, then, $\|f(x)\|_\infty$ will be distributed according to

$$\mathbb{P}\left(\left\|f(x)\right\|_\infty \le y\right) = \prod_{i=1}^m \left[\Phi\left(\frac{y-\mu_i(x)}{\sigma_i(x)}\right) + \Phi\left(\frac{y+\mu_i}{\sigma_i}\right) - 1\right],$$

where Φ is the cumulative distribution function of a standard normal random variable, and $\mu_i(x)$ and $\sigma_i(x)$ denote the posterior mean and standard deviation at x for function component f_i. For a derivation, see Subsection 3.2. Theorem 3.1 then states that the expected improvement for this random variable $\|f(x)\|_\infty$ is given by

$$\mathbb{E}\left[\max\left(f^* - \|f(x)\|_\infty, 0\right)\right] = \int_0^{f^*} \mathbb{P}\left(\left\|f(x)\right\|_\infty \le y\right) \mathrm{d}y.$$

A short proof can be found in Subsection 3.3; here we provide another longer proof.

Proof. if we let Ψ_i and ψ_i denote the cumulative distribution and density function of the absolute values of the components $|f_i|$, respectively, then we can also write the expected improvement as the m-dimensional integral

$$\begin{aligned}\mathrm{EI} &= \int_0^{f^*} \cdots \int_0^{f^*} (f^* - \max_{1\le i\le m} y_i) \prod_{i=1}^m \psi_i(y_i)\, \mathrm{d}y_m \dots \mathrm{d}y_1 \\ &= f^* \prod_{i=1}^m \Psi_i(f^*) - \int_0^{f^*} \cdots \int_0^{f^*} \max_{1\le i\le m} y_i \prod_{i=1}^m \psi_i(y_i)\, \mathrm{d}y_m \dots \mathrm{d}y_1. \end{aligned} \tag{15}$$

We will use induction to rewrite this last integral. For $1 \leq n \leq m$ define

$$g_n := \int_0^{f^*} \cdots \int_0^{f^*} \max_{1\leq i\leq n} y_i \prod_{i=1}^{n} \psi_i(y_i) \prod_{i=n+1}^{m} \Psi_i(\max_{1\leq i\leq n} y_i)\, \mathrm{d}y_n \ldots \mathrm{d}y_1,$$

where we adopt the convention that the empty product $\prod_{i=m+1}^{m} \Psi_i(\max_{1\leq i\leq m} y_i) = 1$. Our induction hypothesis is that

$$g_n = g_{n-1} + \int_0^{f^*} y\psi_n(y) \prod_{\substack{i=1 \\ i\neq n}}^{m} \Psi_i(y)\, \mathrm{d}y \quad (2 \leq n \leq m).$$

To prove this hypothesis, distinguish two cases:

$$\max_{1\leq i\leq n} y_i = \begin{cases} y_n & \text{if } y_n \geq \max\limits_{1\leq i\leq n-1} y_i, \text{ and} \\ \max\limits_{1\leq i\leq n-1} y_i & \text{if } y_n \leq \max\limits_{1\leq i\leq n-1} y_i. \end{cases}$$

We now split the innermost integral in g_n based on these two cases. In the fist case, we find

$$\begin{aligned} &\int_0^{f^*} \cdots \int_0^{f^*} \int_{\max\limits_{1\leq i\leq n-1} y_i}^{f^*} y_n \prod_{i=1}^{n} \psi_i(y_i) \prod_{i=n+1}^{m} \Psi_i(y_n)\, \mathrm{d}y_n \ldots \mathrm{d}y_1 = \\ &\int_0^{f^*} \int_0^{y_n} \cdots \int_0^{y_n} y_n \prod_{i=1}^{n} \psi_i(y_i) \prod_{i=n+1}^{m} \Psi_i(y_n)\, \mathrm{d}y_1 \ldots \mathrm{d}y_n = \\ &\int_0^{f^*} y\psi_n(y) \prod_{\substack{i=1 \\ i\neq n}}^{m} \Psi_i(y)\, \mathrm{d}y, \end{aligned}$$

where the first step was made by changing the order of integration to integrate y_n last. This is the added integral in our induction hypothesis.

In the second case, we find

$$\begin{aligned} &\int_0^{f^*} \cdots \int_0^{f^*} \int_0^{\max\limits_{1\leq i\leq n-1} y_i} \max_{1\leq i\leq n-1} y_i \prod_{i=1}^{n} \psi_i(y_i) \prod_{i=n+1}^{m} \Psi_i(\max_{1\leq i\leq n-1} y_i)\, \mathrm{d}y_n \ldots \mathrm{d}y_1 = \\ &\int_0^{f^*} \cdots \int_0^{f^*} \max_{1\leq i\leq n-1} y_i \prod_{i=1}^{n-1} \psi_i(y_i) \prod_{i=n}^{m} \Psi_i(\max_{1\leq i\leq n-1} y_i)\, \mathrm{d}y_{n-1} \ldots \mathrm{d}y_1 = g_{n-1}. \end{aligned}$$

This proves the induction hypothesis. Furthermore, notice that g_m is equal to the integral in equation (15). We can therefore apply induction to find that this integral is equal to

$$
\begin{aligned}
&g_1 + \sum_{n=2}^{m} \int_0^{f^*} y\psi_n(y) \prod_{\substack{i=1 \\ i\neq n}}^{m} \Psi_i(y)\,\mathrm{d}y \\
&= \int_0^{f^*} y \sum_{n=1}^{m} \psi_n(y) \prod_{\substack{i=1 \\ i\neq n}}^{m} \Psi_i(y)\,\mathrm{d}y \\
&= \int_0^{f^*} y \frac{\mathrm{d}}{\mathrm{d}y}\Big[\prod_{i=1}^{m} \Psi_i(y)\Big]\,\mathrm{d}y.
\end{aligned}
$$

Hopefully, this last integral looks familiar. The function we are differentiating inside the brackets is actually the cumulative distribution function of the max-norm of the components of f! We now apply partial integration and use the result in equation (15) to find that

$$
\mathrm{EI} = f^* \prod_{i=1}^{m} \Psi_i(f^*) - f^* \prod_{i=1}^{m} \Psi_i(f^*) + \int_0^{f^*} \prod_{i=1}^{m} \Psi_i(y)\,\mathrm{d}y.
$$

□

A.3. Properties of the maximum of independent, identically distributed Gaussians

In this subsection, we make a start on deriving properties of the distribution of the maximum-absolute value norm $\|f(\cdot)\|_\infty$, where $f : \mathbb{R}^d \supset \Omega \to \mathbb{R}^m$ is some multi-output function. Under the model assumptions of Bayesian Optimisation, we allocate a Gaussian Process $f_i(x) \sim \mathcal{N}\left(\mu_i(x), \sigma_i^2(x)\right)$ $(x \in \Omega)$ to each function component f_i. As a result, the distribution of $\|f(x)\|_\infty$ is given by Equation (8).
We start in a simplified setting by considering the maximum $m_n := \max_{i=1,\dots,n} \epsilon_i$ of n i.i.d. Gaussians $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. The cumulative distribution function of m_n and the resulting density function are given by

$$
F_{m_n}(z) = \mathbb{P}(\epsilon_1 \leq z)^n \text{ and } f_{m_n}(z) = n f_{\epsilon_1}(z) \mathbb{P}(\epsilon_1 \leq z)^{n-1}.
$$

We can try to solve the resulting intdwwegral for the expected value by partial integration. If we use the shorthand $f(\cdot)$ and $F(\cdot)$ for the density and cumulative distribution function of ϵ_1, respectively, we find that

$$
\begin{aligned}
\mathbb{E}[m_n] &= \int_{-\infty}^{\infty} nzf(z)F(z)^{n-1}\,\mathrm{d}z \\
&= -n\sigma^2 \int_{-\infty}^{\infty} \frac{\mathrm{d}f}{\mathrm{d}z} F(z)^{n-1}\,\mathrm{d}z \\
&= n(n-1)\sigma^2 \int_{-\infty}^{\infty} f(z)^2 F(z)^{n-2}\,\mathrm{d}z. \qquad (16)
\end{aligned}
$$

Here the first step was made by the identity

$$-\sigma^2 \frac{\mathrm{d}f}{\mathrm{d}z} = -\sigma^2 \frac{\mathrm{d}}{\mathrm{d}z}\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\Big(-\frac{z^2}{2\sigma^2}\Big)\right] = zf(z),$$

and the second step by partial integration. The boundary terms vanish because

$$\lim_{z\to\pm\infty} f(z)F(z)^{n-1} = 0$$

for $n \geq 1$. For the case $n = 2$, we can recognise that

$$f(z)^2 = \frac{1}{2\pi\sigma^2}\exp\Big(-\frac{z^2}{\sigma^2}\Big) = \frac{1}{2\sqrt{\pi\sigma^2}}\frac{1}{\sqrt{\pi\sigma^2}}\exp\Big(-\frac{z^2}{\sigma^2}\Big) = \frac{1}{2\sqrt{\pi\sigma^2}}g(z),$$

where $g(z)$ is the density of a $\mathcal{N}(0, \frac{\sigma^2}{2})$ random variable. Therefore we find in this case that

$$\mathbb{E}[m_2] = \frac{2(2-1)\sigma^2}{2\sqrt{\pi\sigma^2}}\int_{-\infty}^{\infty} g(z)\,\mathrm{d}z = \frac{\sigma}{\sqrt{\pi}}. \tag{17}$$

Writing $f(z)^2$ in terms of the density function $g(z)$ has another benefit: we also see that in general, the expected value can be written as

$$\mathbb{E}[m_n] = \frac{n(n-1)\sigma}{2\sqrt{\pi}}\int_{-\infty}^{\infty} F(z)^{n-2}g(z)\,\mathrm{d}z.$$

The question that arises is 'How are the cumulative distribution functions $F(z)$ and $G(z)$ related?'. To answer this, let's dive back into the mathematics:

$$G(z) = \int_{-\infty}^{z} g(t)\,\mathrm{d}t = \int_{-\infty}^{z} \frac{1}{\sqrt{\pi\sigma^2}}\exp\Big(-\frac{t^2}{\sigma^2}\Big)\,\mathrm{d}t$$

$$t \mapsto \frac{t}{\sqrt{2}} \implies \int_{-\infty}^{\sqrt{2}z} \frac{1}{\sqrt{\pi\sigma^2}}\exp\Big(-\frac{t^2}{2\sigma^2}\Big)\frac{1}{\sqrt{2}}\,\mathrm{d}t = \int_{-\infty}^{\sqrt{2}z} f(t)\,\mathrm{d}t = F(\sqrt{2}z),$$

or, in other words, $F(z) = G(z/\sqrt{2})$. This implies that the expected value of the maximum is given by

$$\mathbb{E}[m_n] = \frac{n(n-1)\sigma}{2\sqrt{\pi}}\int_{-\infty}^{\infty} G\left(\frac{z}{\sqrt{2}}\right)^{n-2} g(z)\,\mathrm{d}z.$$

Unfortunately, this is as far as we can go analytically. If it had been $G(z)$ instead of $G(z/\sqrt{2})$ we could have used the substitution $t = G(z)$, but that does not work here.