BAYESIAN OPTIMIZATION VIA NORMALIZING FLOWS (FLOWBO)

Contents

Abstract

Bayesian Optimization (BO) is a powerful framework for optimizing expensive black-box functions, combining surrogate models with acquisition functions to guide sampling. This dissertation focuses on Gaussian Processes as surrogates, emphasizing their uncertainty modeling and kernel design, and addresses the challenge of acquisition function scalability in high dimensions. We propose a novel BO method that replaces the acquisition function with a normalizing flow framework, using *local Gaussian Processes* to learn probability distributions over promising points. Experiments show that our approach scales more efficiently with dimension and achieves faster iterations than standard BO, though it tends towards over-exploitation due to limitations of the local models. We outline strategies for improvement, including adaptive local modeling, geometry-aware flows, and spline-based transformations, to enhance robustness in high-dimensional optimization.

Chapter 1

Introduction

In many scientific and engineering domains, optimizing complex systems often involves evaluating objective functions that are expensive, time-consuming, or otherwise impractical to measure exhaustively. For instance, tuning hyperparameters in deep neural networks can require training models for hours or days; designing aerodynamic shapes involves costly simulations or physical experiments; and optimizing chemical processes demands precious laboratory resources. In such settings, traditional optimization methods, such as grid search or random search, become prohibitively inefficient. Bayesian Optimization (BO) offers a principled and efficient alternative by building surrogate models that approximate the objective function, thereby guiding the search towards promising regions with fewer evaluations.

This dissertation introduces the theory and practice of Bayesian Optimization, with a particular focus on Gaussian Processes (Section 2.2), a flexible class of surrogate models that provide not only predictions but also uncertainty estimates essential for informed decision-making. Following this, we explore acquisition functions (Section 2.2.4), the second critical component of BO, which balance the exploration of unknown regions and the exploitation of areas predicted to have high reward. Despite their success, acquisition functions often suffer from scalability issues in high-dimensional spaces (Section 2.2.5), presenting notable challenges.

To address these issues, in Section 3 we present a novel model that replaces the traditional acquisition function with a framework based on normalizing flows (Section 2.3). This new approach scales well to higher-dimensional problems and aims to improve sampling efficiency by learning a probability distribution over promising evaluation points. Section 4 benchmarks our model against standard approaches, comparing speed, performance, and robustness across varying numbers of dimensions. Finally, Section 5 critically assesses potential shortcomings of our method and outlines avenues for future research.

Chapter 2

Preliminaries

2.1 Bayesian Inference

2.1.1 Overview

Bayesian Optimization (BO) [23, 33, 6] is a powerful framework for optimizing objective functions that are expensive to evaluate and lack an explicit analytical form. Formally, it seeks to solve the global optimization problem:

$$\max_{x \in \mathcal{A}} f(x), \tag{2.1}$$

where $\mathcal{A} \subseteq \mathbb{R}^d$ is the feasible domain and f is a black-box objective function. At its core, Bayesian Optimization addresses the challenge of optimizing functions for which each evaluation is costly; whether in terms of computation time, financial expense, or experimental effort and where only a limited number of such evaluations can be performed.

Critically, the function f is assumed to be inaccessible in closed form, and its values can only be queried pointwise, potentially with noise. The goal is to find an approximate global maximum of f using as few evaluations as possible. This is particularly relevant in applications such as hyperparameter tuning of machine learning models [34] or automated discovery in scientific domains like materials and molecule design [13], where each evaluation (e.g., a simulation or lab experiment) can take hours or incur substantial financial or opportunity costs.

The typical characteristics of BO problems include [7]:

- **Low-dimensional input space:** The input x lies in $\mathbb{R}^d$ for relatively small d, often with $d \leq 20$ in most practical applications. Bayesian methods tend to struggle in high-dimensional settings due to the curse of dimensionality.

- **Continuity of the objective function**: This assumption is crucial when we come to modelling f.
- **Expensive evaluations:** The function f is costly to evaluate, which imposes a tight budget on the number of function queries. For instance, in molecular discovery, evaluating f might correspond to a high-fidelity chemistry simulation or a physical laboratory assay, each of which may take hours or incur substantial financial or opportunity costs.
- **Derivative-free observations:** Only the function values $f(x)$ are observed at each query point, without access to gradients or higher-order derivatives. As a result, traditional optimization techniques such as gradient descent, Newton's method, or quasi-Newton methods are not applicable. Problems of this type are commonly referred to as *derivative-free optimization* problems and are explored more in [21].

2.1.2 History and Motivation

Bayesian optimization (BO) is a statistical approach to experimental design that treats each experiment as a purposeful step toward learning, rather than just collecting data. The core idea is to balance what is already known about the system with the uncertainty that remains, guiding the choice of future experiments. BO works by selecting experiments sequentially, each time using the results of past evaluations to more efficiently search for the global optimum of the function being studied. This makes BO a principled and efficient method for optimizing expensive, black-box functions where evaluations are limited.

Experimental design itself is a longstanding discipline in statistics, dedicated to the optimal planning of experiments so that robust inferences can be drawn with maximum efficiency and minimal resource expenditure. The field traces its formal origins back more than two centuries [9, 27], but foundational contributions emerged in the early 20th century. Smith's 1918 dissertation set the groundwork by considering how to allocate experiments for polynomial regression in a way that minimized predictive uncertainty. Fisher's groundbreaking research at the Rothamsted Experimental Station soon followed, playing a pivotal role in modernizing statistical experimentation by establishing rigorous, systematic approaches for designing and analyzing experiments. These foundational developments gave rise to the field of "optimal design".

Initial work in optimal design was generally concerned with fixed, non-adaptive designs — that is, selecting all experimental conditions in advance to minimize some global measure of uncertainty. However, these early approaches did not address the sequential nature of real-world decision-making, where each experimental result could inform the strategy for the next.

A transformative shift occurred during World War II, when Abraham Wald founded the field of sequential analysis. Wald showed that experiments could be adaptively terminated or continued based on interim results, often yielding dramatic savings in resources and time. The practical

value of this work was immediately recognized — it was classified during the war and published only later [39, 40].

Friedman and Savage [8] advanced these ideas by proposing sequential strategies for designing experiments explicitly aimed at finding a maximum. They criticized the inefficiency of non-adaptive procedures, arguing that adaptivity reduces wasted effort in unpromising regions and accelerates optimization progress.

Kushner synthesized Bayesian statistics and sequential design, introducing a formal framework for optimizing noisy unknown functions. Kushner's contributions integrated a Gaussian process surrogate model and leveraged Bayesian decision theory to select experiments aimed at maximizing the objective function. His work was the first to introduce Bayesian optimization policies: maximizing an upper confidence bound and maximizing probability of improvement.

More recently, Snoek [34] observed that BO was useful for hyperparameter training in deep neural networks, leading to a surge of interest in the machine learning community. This led to further developments in reinforcement learning [36], multi-fidelity hyperparameter tuning [43], and neural architecture search [18].

2.2 Gaussian Processes

2.2.1 Definition and Basics

Since BO aims to maximise f using a small number of evaluations, we need a way to make informed guesses about its values at points that have not yet been evaluated. In Bayesian Optimization, this role is fulfilled by the *surrogate model*. Crucially, the surrogate model does not just predict the value of the function, it also quantifies its own uncertainty by providing a posterior distribution of the possible values of $f(x)$ at each point.

The surrogate model for f is most commonly chosen to be a Gaussian Process - an extension of the familiar Multivariate Normal Distribution (MVN) now supported over an arbitrary infinite domain $\mathcal{X}$.

Definition 2.1 (Gaussian Process [29])**.** A *Gaussian Process* (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Just like with the MVN, a Gaussian Process is fully specified by its mean and covariance. However, we must replace the finite-dimensional parameters of the MVN by corresponding *functions* over the required space. We thus define the mean function $\mu : \mathcal{X} \to \mathbb{R}$ and covariance function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as

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

$$K(x, x') = \mathbb{E}\Big[(f(x) - \mu(x))(f(x') - \mu(x'))\Big], \tag{2.3}$$

and we write the Gaussian process as

$$f(x) \sim \mathcal{GP}\Big(\mu(x), K(x, x')\Big). \tag{2.4}$$

With the mean and covariance functions defined we may now find the joint distribution of a finite number of points, say $\mathbf{x} = \{x_1, x_2, \ldots, x_n\} \subset \mathcal{X}$ with corresponding function values $\boldsymbol{\phi} = f(\mathbf{x})$. The Gaussian Process implies that the vector of function values $\boldsymbol{\phi} = [f(x_1), f(x_2), \ldots, f(x_n)]^\top$ follows a MVN with mean vector and covariance matrix constructed from the mean and covariance functions evaluated at the inputs in $\mathbf{x}$.

Formally,

$$\boldsymbol{\phi} \sim \mathcal{N}\left(\begin{bmatrix}\mu(x_1)\\ \mu(x_2)\\ \vdots \\ \mu(x_n)\end{bmatrix}, \mathbf{K}\right), \tag{2.5}$$

where the covariance matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ is defined by

$$\mathbf{K}_{ij} = K(x_i, x_j) \quad \text{for } i, j = 1, \ldots, n. \tag{2.6}$$

This covariance matrix $\mathbf{K}$ is often referred to as the **Gram matrix** (or kernel matrix), which encodes the pairwise covariances between the function values at the input points in $\mathbf{x}$. The Gram matrix is symmetric and positive semi-definite by construction, reflecting the properties of the covariance function.

The structure of $\mathbf{K}$ is central to computations with Gaussian Processes, as it determines the shape of the prior over functions and is used extensively in predictive inference and model updating.

Marginalization A fundamental property of Gaussian Processes that ensures their coherence as priors over functions is the *consistency* or *marginalization property*. This property guarantees global consistency of the model's beliefs across different sets of input points. More precisely, consider an arbitrary finite set of points $\mathbf{x}$, and a superset $\mathbf{x}' \supset \mathbf{x}$. The marginal distribution of the function values associated with $\mathbf{x}$, denoted $\boldsymbol{\phi} = f(\mathbf{x})$, remains identical whether it is computed directly from the Gaussian Process prior restricted to $\mathbf{x}$, or indirectly by first considering the joint distribution over the larger set $\mathbf{x}'$ and then marginalizing out the additional points $\mathbf{x}' \setminus \mathbf{x}$. This ensures that the GP defines a family of finite-dimensional distributions that are *mutually*

consistent, a requirement formalized by the Kolmogorov extension theorem [45].

As an example, suppose we have a vector of function values $\boldsymbol{\phi} = [f(x_1), \ldots, f(x_n)]^\top$ drawn from a GP with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{K}$. If we take any subset of indices $I \subseteq \{1, \ldots, n\}$, the corresponding subset of function values $\boldsymbol{\phi}_I$ also follows a normal distribution:

$$\boldsymbol{\phi}_I \sim \mathcal{N}(\boldsymbol{\mu}_I, \mathbf{K}_{II}),$$

where $\boldsymbol{\mu}_I$ and $\mathbf{K}_{II}$ are the mean vector and covariance submatrix for the subset. This means the distribution over any subset is consistent with the full distribution, ensuring the GP is a coherent model over functions. As for the simplest case, consider the set $\mathbf{x}$ to be a single point $x \in \mathcal{X}$. The marginal distribution of the function value $f(x)$ at this point must be Gaussian with mean $\mu(x)$ and variance $K(x, x)$ inherited from the GP's mean and covariance functions.

Example of Sample Functions An example of such a Gaussian Process is given in Figure 2.1. Here we have used a Gaussian Process prior with a zero mean function and a Radial Basis Function (RBF) kernel (detailed in Section 2.2.3) to generate samples. To approximate the samples, we use a dense grid of 1000 points equally spaced over the interval. The plot illustrates the diversity of plausible functions under the GP prior, its inherent smoothness due to the RBF kernel, and the associated uncertainty at each point. The model's ability to quantify uncertainty is what makes it particularly valuable for selecting optimal points in experimental design. We now see how the model adapts with observations of the function.

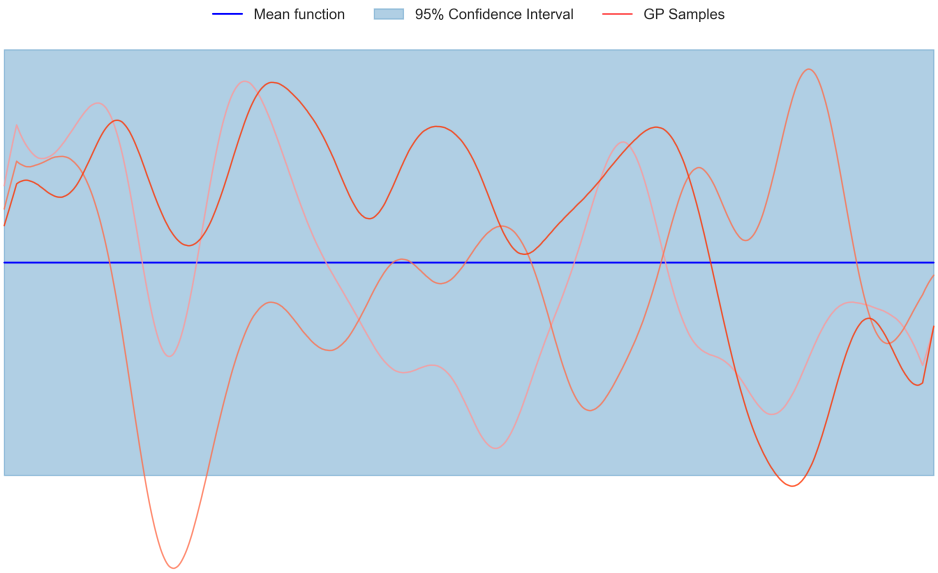


Figure 2.1: Samples drawn from a Gaussian Process prior with a zero mean and RBF kernel. The input domain is restricted to the range $[5, 20]$. Three sample functions are plotted along with the posterior mean and the 95% confidence interval. We note the high degree of flexibility visible in these samples, making them ideal for approximating complex objective functions.

2.2.2 Gaussian Process Inference

Inference in the context of Gaussian Processes refers to the process of updating our beliefs about an unknown function after observing its values at certain input locations. We begin with a Gaussian Process prior over functions f:

$$p(f) = \mathcal{GP}(f; \mu, K),$$

where $\mu : \mathcal{X} \to \mathbb{R}$ is the mean function and $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the covariance kernel.

Suppose we observe data $D = \mathbf{y}$, which shares a joint Gaussian Process distribution with f, characterized by:

$$p\begin{pmatrix} f \\ \mathbf{y} \end{pmatrix} = \mathcal{GP}\left(\begin{bmatrix} f \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} \mu \\ \mathbf{m} \end{bmatrix}, \begin{bmatrix} K & \kappa^\top \\ \kappa & C \end{bmatrix}\right). \tag{2.7}$$

Here, μ denotes the prior mean function evaluated at arbitrary input locations, while $\mathbf{m} = \mu(\mathbf{x}_{\text{obs}})$ represents the vector of prior mean values evaluated specifically at the observed input points $\mathbf{x}_{\text{obs}}$. The function $K = K(\mathbf{x}, \mathbf{x}')$ is the prior covariance function (or kernel) evaluated at pairs of arbitrary input points, describing the covariance between function values at those locations. The matrix $\kappa = K(\mathbf{x}_{\text{obs}}, \mathbf{x})$ contains the cross-covariances between the observed inputs $\mathbf{x}_{\text{obs}}$ and the arbitrary inputs $\mathbf{x}$. Lastly, $C = K(\mathbf{x}_{\text{obs}}, \mathbf{x}_{\text{obs}}) + \Sigma_{\text{noise}}$ is the covariance matrix computed over the observed inputs, where Σ_{noise} represents the noise covariance included if the observations are noisy; if the data is noise-free, Σ_{noise} is zero.

Conditioning this joint Gaussian Process on the observed data $\mathbf{y}$ gives the posterior Gaussian Process:

$$p(f \mid D) = \mathcal{GP}\left(f; \mu_D, K_D\right),$$

with posterior mean function

$$\mu_D(\mathbf{x}) = \mu(\mathbf{x}) + \kappa(\mathbf{x})^\top C^{-1}\Big(\mathbf{y} - \mathbf{m}\Big)$$

and posterior covariance function

$$K_D(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - \kappa(\mathbf{x})^\top C^{-1}\kappa(\mathbf{x}').$$

Thus, after observing $\mathbf{y}$, the posterior retains the Gaussian Process structure but with updated mean and covariance functions reflecting the information gained from the data.

We present an example of updating a GP in Figure 2.2 after observing some data points. As observations are added, the GP posterior mean is updated to pass near the observed points, effectively interpolating the data while maintaining smoothness imposed by the prior kernel. The

posterior covariance, which defines the uncertainty around the mean, contracts near the observed points, reflecting increased confidence where data are available, and remains broader in regions where the function is still unobserved. In this case our observations are noise free and hence there is zero uncertainty at the observed points - this corresponds to setting Σ_{noise} to zero.

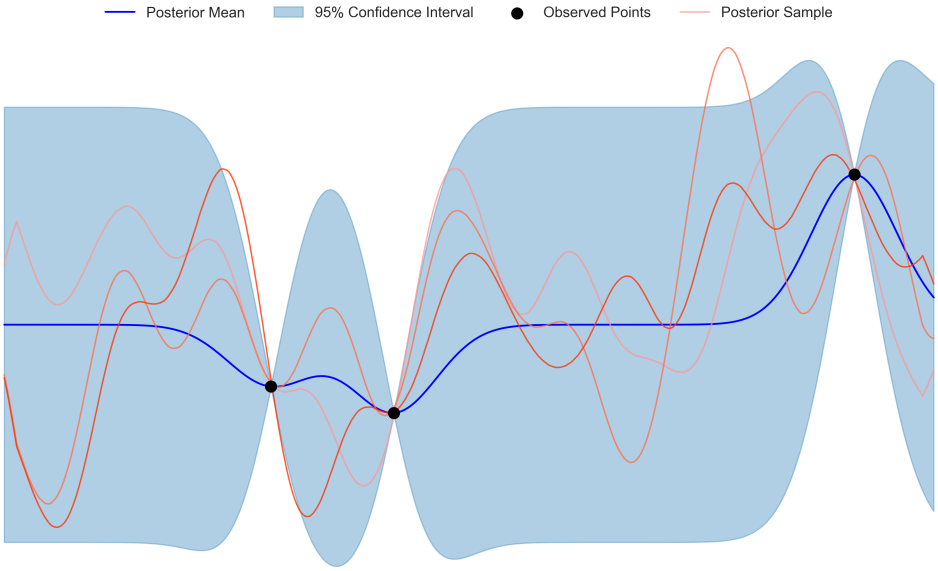


Figure 2.2: Posterior distribution of a Gaussian Process conditioned on three observed points within the input domain $[5, 20]$. The black dots indicate the observed data points used for conditioning. This illustrates how the Gaussian Process posterior adapts and reduces uncertainty near the observed locations while expressing higher uncertainty elsewhere.

2.2.3 Prior Mean and Covariance Functions

Prior Mean Function The prior mean function in a Gaussian Process represents our baseline expectation of the function before seeing any data. It plays its most significant role in extrapolatory regions, far away from observed data, where the posterior is influenced primarily by this prior belief. Importantly, the mean function affects the GP distribution only up to an additive translation, shifting the expected value of the function but not altering its shape or covariance properties. However, using a non-constant or overly complex mean function can be risky, as it may introduce unintended biases or overly rigid assumptions that dominate the inference, especially in sparsely observed regions, potentially leading to misleading predictions. For these reasons, particularly in applications like Bayesian Optimization where flexibility and robustness are key, the mean function is most often chosen to be a constant (often zero or the empirical mean of observed data). This simplicity prevents the model from being unduly influenced by prior assumptions and has been supported empirically [4].

Covariance Function

The covariance function (also called the *kernel*) is arguably the most critical component in a Gaussian Process model. While the mean function only shifts the process and is mainly relevant

in unobserved (extrapolatory) regions, the covariance function defines the structure of correlations between function values at different input locations. In effect, the choice of covariance function determines the smoothness, periodicity, and general behavior of the random functions generated by the GP. It is the kernel, not the mean, that encodes our assumptions about the function class—such as whether outputs at similar inputs should co-vary strongly, and on what scale. We see impact of the mean and covariance on GP samples in Figure 2.3.

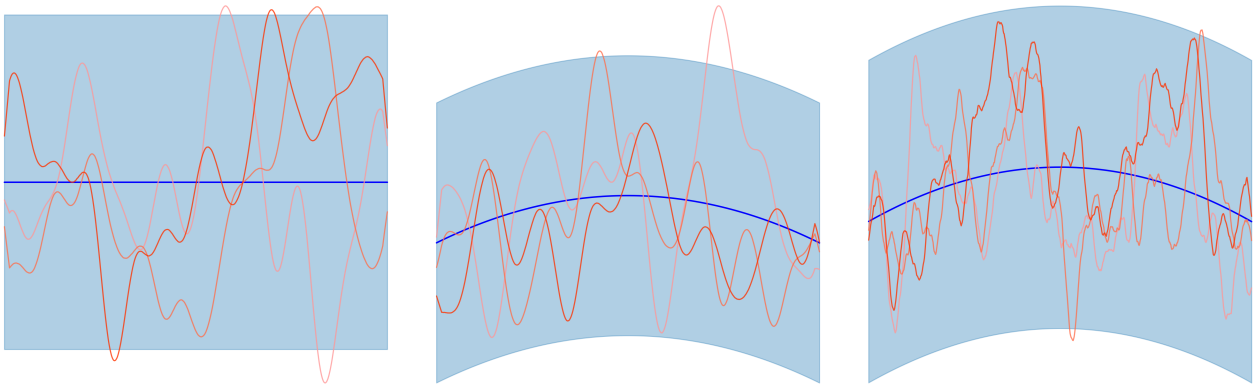


Figure 2.3: Comparison of Gaussian Process posterior plots: The left plot shows a zero mean function with an RBF kernel producing smooth functions. The middle plot illustrates a quadratic mean function with the same RBF kernel, creating a slight curvature in the mean. The right plot features the quadratic mean combined with a Matern kernel, resulting in more jagged sample functions due to the less smooth covariance function. We see that the covariance function has a much larger impact on the samples than the mean.

Mathematically, the covariance function is defined by:

$$K(x, x') = \text{cov}[\phi, \phi' \mid x, x'],$$

where ϕ, ϕ' are the random function values at x and x', respectively. This kernel function quantifies how similar the outputs are expected to be, given inputs x and x'; the more similar the two inputs under the kernel, the stronger their statistical dependence.

A covariance function must satisfy certain mathematical criteria. It must be *symmetric*:

$$K(x, x') = K(x', x)$$

and, most importantly, it must generate a *positive semidefinite* Gram matrix for any finite set of input points. This means that all eigenvalues of the Gram matrix are non-negative, ensuring that the resulting joint Gaussian distribution is valid. Formally, a kernel k is said to be positive semidefinite if, for all square-integrable functions f over the input space X with respect to measure μ,

$$\int \int k(x, x') f(x) f(x') \, d\mu(x) \, d\mu(x') \geq 0. \tag{2.8}$$

This property guarantees that any linear combination of function values will not produce negative

variance.

Some covariance functions possess special properties; for example, a *stationary* covariance function depends only on the distance between inputs, not on their absolute location. That is,

$$K(x, x') = k(x - x').$$

Stationary kernels are popular because they express the assumption of uniform behavior throughout the domain. We explore two examples of such kernels below.

Squared Exponential (SE) / Radial Basis Function (RBF) Kernel: The most widely used stationary kernel is the squared exponential (SE), also known as the Gaussian or radial basis function (RBF) kernel, defined as

$$k_{\text{SE}}(r) = \exp\left(-\frac{r^2}{2\ell^2}\right),$$

where $r = \|x - x'\|$ and ℓ is the characteristic length-scale. The length-scale controls how quickly correlations decay; smaller ℓ means only nearby points are strongly correlated. The SE kernel is infinitely differentiable, so sample paths drawn from a GP with this kernel are extremely smooth. Due to its flexibility and smoothness, the RBF kernel is prevalent throughout the kernel methods literature.

However, despite its popularity, some researchers - statistician Michael Stein [35] in particular - have cautioned against using the squared exponential kernel in practice, as its excessive smoothness may not reflect the true underlying process in real-world data. Stein encourages practitioners to use more flexible kernels, like the Matérn class, which allows explicit control over smoothness.

Matérn Kernel: The Matérn kernel is a generalization that introduces a smoothness parameter ν controlling the differentiability of the process:

$$k_{\text{Matérn}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\, r}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}\, r}{\ell}\right),$$

where K_ν is the modified Bessel function of the second kind, ℓ is the length-scale, and $\nu > 0$. As ν increases, the kernel (and thus the GP samples) become smoother: for half-integer values $\nu + 0.5 \in \mathbb{N}$, the process is $\lfloor \nu \rfloor$-times mean-square differentiable. For example, $\nu = 1/2$ gives the exponential kernel with nondifferentiable sample paths; $\nu = 3/2$ and $\nu = 5/2$ produce once and twice mean-square differentiable functions, respectively. In the limit as $\nu \to \infty$, the Matérn kernel converges to the squared exponential kernel.

Marginal Likelihood In Gaussian Processes, the *hyperparameters* $\boldsymbol{\theta}$ are those parameters that define the covariance function (kernel), mean function, and observation model — most notably, length scales in the kernel and the observation noise variance Σ_{noise}. These hyperparameters govern the properties of the prior over functions $f(x)$, denoted $\boldsymbol{\phi}(x) = f(x)$, but are distinct from the latent function values themselves.

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, the posterior distribution over the hyperparameters is obtained using Bayes' theorem:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) \propto p(\boldsymbol{\theta})\, p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}),$$

where $p(\boldsymbol{\theta})$ is the prior over hyperparameters, often chosen to be uninformative (uniform), i.e., $p(\boldsymbol{\theta}) \propto 1$. The term

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\phi}, \boldsymbol{\theta})\, p(\boldsymbol{\phi} \mid \mathbf{X}, \boldsymbol{\theta})\, d\boldsymbol{\phi}$$

is known as the *marginal likelihood* or *model evidence*, representing the likelihood of the observed data marginalized over all latent function values $\boldsymbol{\phi}$.

Thanks to the Gaussian assumptions in GPs, this integral admits the closed form

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}\Big(\mathbf{y}; \boldsymbol{\mu}, \mathbf{K} + \sigma_n^2 \mathbf{I}\Big),$$

where $\boldsymbol{\mu}$ is the mean vector, $\mathbf{K}$ is the covariance matrix computed with kernel parameters $\boldsymbol{\theta}$, and σ_n^2 is the observation noise variance.

The corresponding log marginal likelihood is given by:

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \underbrace{(\mathbf{y} - \boldsymbol{\mu})^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} (\mathbf{y} - \boldsymbol{\mu})}_{\text{fit term}} - \frac{1}{2} \underbrace{\log\det(\mathbf{K} + \sigma_n^2 \mathbf{I})}_{\text{complexity penalty}} - \frac{n}{2} \log 2\pi. \tag{2.9}$$

This expression automatically balances model fit and complexity [22]: the *fit term* measures how well the GP explains the observed data while the *complexity penalty* pushes for a simple model.

2.2.4 Acquisition Functions

So far we have introduced Gaussian Processes as a powerful, probabilistic surrogate for the black-box function in Equation (2.1). Having established this probabilistic representation, the next step is to guide the search for new evaluations in a principled way. Bayesian decision theory offers a rigorous framework to achieve this by formalizing decision making under uncertainty. Central to this framework is the concept of an optimization policy, which is a strategy or rule that determines the choice of the next input point(s) to evaluate based on current knowledge and uncertainty. This leads to a sequential strategy that iteratively proposes new points to evaluate, observes the outcomes, updates the GP posterior, and refines the belief over the function (see Algorithm 1).

Algorithm 1 Bayesian Optimization Procedure for Maximization

Input: Objective function f, budget N, initial sample size n_0
Initialize: Select n_0 initial points $\{\mathbf{x}_1, \ldots, \mathbf{x}_{n_0}\}$
Observe responses $y_i = f(\mathbf{x}_i)$ for $i = 1, \ldots, n_0$
Fit a Gaussian Process model to $(\mathbf{x}_i, y_i)_{i=1}^{n_0}$
for $n = n_0 + 1$ **to** N **do**
    Update the Gaussian Process posterior with all collected data
    Optimize the acquisition function to select the next query point $\mathbf{x}_n$
    Augment the dataset with $(\mathbf{x}_n, y_n = f(\mathbf{x}_n))$
end for
Output: Return $\mathbf{x}_{\text{best}}$, where $y_{\text{best}} = \max\{y_1, \ldots, y_N\}$, or the point with the highest posterior mean

To quantify and compare the desirability of different outcomes, we introduce a *utility function* $u(a, \psi)$, where a denotes the action taken and ψ denotes the unknown state of the world (for instance, the value of the black-box function at a query point). The utility function captures our preferences: higher values represent more preferred outcomes.

However, because the true state ψ is unknown and only captured probabilistically via a GP, we cannot maximize utility directly. Instead, Bayesian theory directs us to maximize the *expected utility* (acquisiton function):

$$a \in \arg\max_{a' \in \mathcal{A}} \mathbb{E}[u(a', \psi) \mid D]$$

where D is the observed data. According to von Neumann and Morgenstern [38], maximizing expected utility is not just an intuitive strategy; it is, under certain rationality axioms, the only coherent (internally consistent) way to make decisions under uncertainty. Their axiomatic approach establishes that if a decision maker's preferences satisfy some basic principles (such as completeness, transitivity, independence, and continuity), then their behavior can be represented as maximizing the expected value of some utility function.

Probability of Improvement One of the simplest acquisition functions is the Probability of Improvement (PI). It is motivated by the desire to sample at points most likely to improve upon the current best observed value y^*. The utility function for PI is

$$u_{\text{PI}}(y) = \begin{cases} 1 & \text{if } y > y^* \\ 0 & \text{otherwise} \end{cases}$$

where y^* is the best value observed so far (assuming maximization). This means a reward of one is received for improvement, and nothing otherwise. The corresponding acquisition function is

the probability, under the current Gaussian Process posterior, that sampling at x will yield a new best:

$$\alpha_{\mathrm{PI}}(x) = \mathbb{E}[u_{\mathrm{PI}}(y)] = P(y > y^* \mid x, D)$$

In practical terms, maximizing PI encourages choosing points most likely to yield *any* improvement, but ignores the *magnitude* of the improvement. Thus, PI can be overly myopic and is prone to getting stuck in local minima.

Expected Improvement The Expected Improvement (EI) acquisition function [16] addresses this limitation by explicitly considering the size of the improvement over the current best observed value. Its utility function is

$$u_{\mathrm{EI}}(y) = \max(0, y^* - y)$$

which rewards outcomes in proportion to how much they improve on the current best. The acquisition function is then

$$\alpha_{\mathrm{EI}}(x) = \mathbb{E}\left[\max(0, f^* - f(x)) \mid x, D\right]$$

where, when the predictive distribution is normal, this has the closed form:

$$\alpha_{\mathrm{EI}}(x) = (f^* - \mu(x))\,\Phi\left(\frac{f^* - \mu(x)}{\sqrt{K(x,x)}}\right) + \sqrt{K(x,x)}\,\mathcal{N}\left(f^*; \mu(x), K(x,x)\right)$$

The first term in the EI acquisition function may be increased by reducing the mean and thus may be viewed as encoding exploitation. The second term can be increased by increasing variance and thus encodes exploration. Thus by striking a balance between the two it avoids the limitations of strategies that are purely greedy or random.

Figure 2.4 illustrates a Gaussian Process regression model alongside the Expected Improvement acquisition function, which guides the selection of the next sampling point. Notably, the acquisition function approaches zero at previously sampled locations, reflecting the low probability of achieving improvement there. Conversely, it attains its highest values near the best observed sample, thereby effectively balancing exploration of uncertain regions with exploitation of promising areas.

2.2.5 High Dimensional Optimization

In modern applications, high-dimensional Bayesian optimization is increasingly required, with one of the most prominent examples being hyperparameter tuning in machine learning models [28].

As the dimensionality of the problem grows, two major challenges arise: (1) fitting the Gaussian Process surrogate itself becomes more difficult, and (2) maximizing the acquisition function

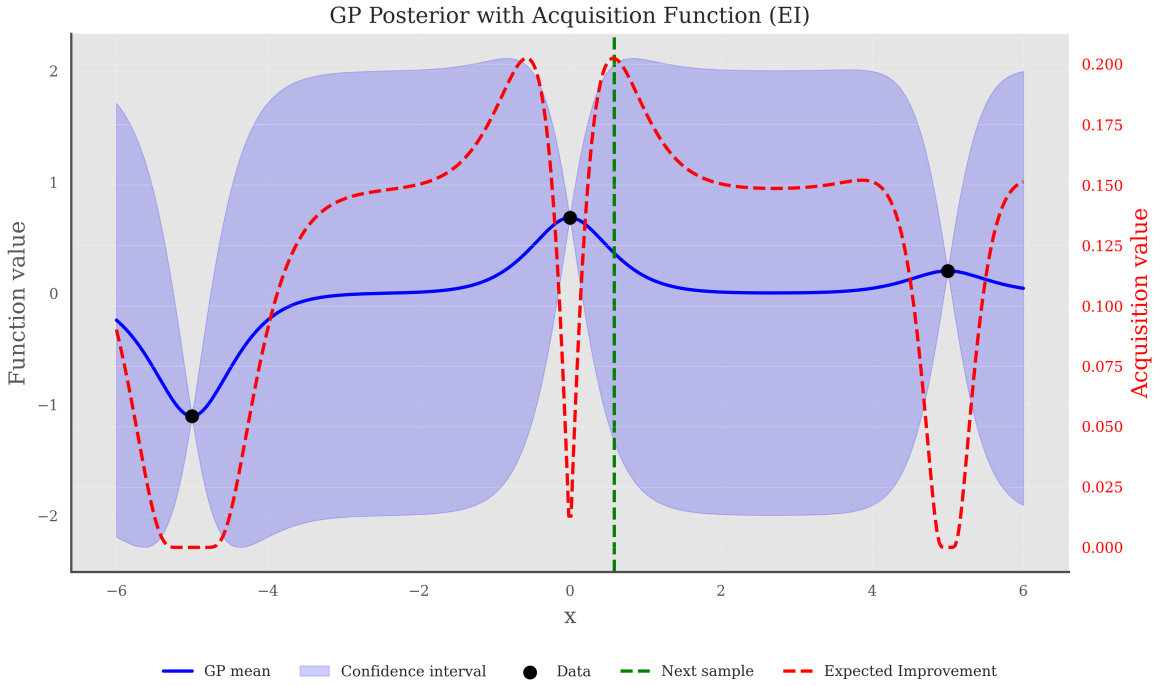


Figure 2.4: We see how the Expected Improvement identifies promising regions near the best observed sample, while still balancing exploration of uncertain areas of the input space. Its value is low at previously sampled points, discouraging redundant evaluations.

(AF) becomes both statistically and computationally intractable. From the perspective of the GP, each additional input dimension introduces new kernel hyperparameters, especially when using automatic relevance determination (ARD), resulting in a parameter space that grows with dimension. To estimate these hyperparameters accurately and avoid overfitting, a much larger amount of data is required, yet evaluating the black-box function often remains expensive or slow. Compounding this, the computational cost of fitting a GP scales cubically with the number of data points, $O(N^3)$.

We further encounter problems with acquired function optimization in high dimensions. AFs tend to possess large, flat plateaus interspersed with sharp local optima [41]. In high dimensions, these plateaus dominate the space, making it challenging to search for global optima efficiently. Gradient-based optimizers such as stochastic gradient descent [19] and L-BFGS-B [44] are easily trapped in sub-optimal regions. Meanwhile, sampling-based approaches suffer acutely from the curse of dimensionality - where the volume of the search space grows exponentially. Furthermore [26] demonstrated that incorrect prior length scales lead to vanishing gradients when maximizing the Marginal Log Likelihood in high dimensions, meaning they remain at the values they were initialized to.

To address these challenges, a common class of solutions aims to reduce the effective dimensionality of the search space before optimizing the acquisition function. One approach is to encode the input variables into a low-dimensional latent space using methods such as Variational Autoencoders

(VAEs) [20, 31, 1, 3]. The acquisition function can then be optimized in the latent feature space, leveraging the assumption that the objective's true variation is primarily in a few directions. Linear projection methods [42, 24] are a typical starting point, but to accommodate more complex structures, geometry-aware Bayesian optimization methods [25, 15] have been proposed, which use nonlinear mappings to uncover low-dimensional manifolds in the data. Yet, these approaches can require substantial data for training and may struggle when the true effective dimensionality is not low.

2.3 Normalizing Flows

Motivated by the large computational overhead of maximizing acquisition functions in high dimensions, we began developing novel and efficient ways to replace the need for them in the Bayesian optimization loop. Before describing our method, we first present some background material on Normalizing Flows - the technique central to our work.

A fundamental problem in statistics and machine learning is to learn the underlying probability distribution of data from observed samples. This task is challenging because the true data distribution is typically unknown and can be complex or high-dimensional. Neural network approaches such as Generative Adversarial Networks (GANs) [11] have been proposed to tackle this challenge by learning to generate realistic data samples through an adversarial training process, where a generator network produces samples and a discriminator network evaluates their realism. Another popular framework is variational inference, which introduces latent variables to explain observed data and optimizes a surrogate posterior distribution to approximate the true hidden structure. Variational Autoencoders (VAEs) [20] are a notable example, combining neural networks with variational inference to learn latent representations and generate data. However, both GANs and VAEs can be difficult to train reliably due to issues like mode collapse and vanishing gradients [2, 32]. In particular, variational inference is often constrained by the choice of the approximate posterior, which must be easy to sample from and compute but also flexible enough to capture the true posterior complexity [12, 37]. Mixture models have been used as richer approximations to the posterior [14, 17, 10], 2012], but these tend to be computationally expensive to train. To overcome these challenges, Rezende and Mohamed [30] utilized normalizing flows, a framework that constructs complex posterior distributions by successively applying a series of invertible and differentiable transformations to a simple base distribution. This approach allows for flexible, tractable density estimation and efficient sampling, significantly improving the representational power of variational inference.

2.3.1 Basics

We begin with a vector $\mathbf{z} \in \mathbb{R}^p$ drawn from a distribution $q_0(\mathbf{z}) : \mathbb{R}^p \to \mathbb{R}$ that is both easy to sample from and tractable to evaluate, such as a standard normal Gaussian. We then apply an

invertible and smooth mapping $\mathbf{g} : \mathbb{R}^p \to \mathbb{R}^p$ to transform $\mathbf{z}$ into another vector $\mathbf{y}$, i.e.,

$$\mathbf{y} = \mathbf{g}(\mathbf{z}),$$

with inverse function $\mathbf{f} = \mathbf{g}^{-1}$. Using the change of variables formula, the density of $\mathbf{y}$ can be expressed as

$$p_{\mathbf{Y}}(\mathbf{y}) = q_0\Big(\mathbf{f}(\mathbf{y})\Big) \left|\det \frac{\partial \mathbf{f}(\mathbf{y})}{\partial \mathbf{y}}\right| = q_0(\mathbf{z}) \left|\det \frac{\partial \mathbf{g}(\mathbf{z})}{\partial \mathbf{z}}\right|^{-1}, \tag{2.10}$$

where the second equality follows from the inverse function theorem. We thus have a way of find the probability of a more complicated distribution as long as g is a smooth invertible map. Thus to model more complex distributions, we may construct a sequence (composition) of simpler invertible transformations $\mathbf{g} = \mathbf{g}_1 \circ \mathbf{g}_2 \circ \cdots \circ \mathbf{g}_K$. Applying the change of variables formula iteratively yields

$$p_{\mathbf{Y}}(\mathbf{y}) = q_0(\mathbf{z}_0) \prod_{k=1}^{K} \left|\det \frac{\partial \mathbf{g}_k(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}}\right|^{-1},$$

where $\mathbf{z}_K = \mathbf{y}$ and $\mathbf{z}_0 = \mathbf{f}(\mathbf{y})$ is obtained by following the inverse transformations

$$\mathbf{z}_0 = \mathbf{f}_1 \circ \mathbf{f}_2 \circ \cdots \circ \mathbf{f}_K(\mathbf{y}),$$

with each $\mathbf{f}_k = \mathbf{g}_k^{-1}$.

Thus, to evaluate the density of a sample $\mathbf{y}$ from the complex distribution, we successively apply the inverse transformations to map $\mathbf{y}$ back to the base variable $\mathbf{z}_0$ whose density $q_0(\mathbf{z}_0)$ is known, and accumulate the Jacobian determinants of each transformation as in the formula above. Most commonly, as is the case in VAEs, the base distribution q_0 is chosen to be a standard normal Gaussian - hence the name "*normalizing* flows."

2.3.2 Radial Flows

Our work extensively uses a family of normalizing flows called radial flows. A radial flow transforms a vector $\mathbf{z} \in \mathbb{R}^p$ using the following invertible mapping:

$$\mathbf{g}(\mathbf{z}) = \mathbf{z} + \beta\, h(\alpha, r)\, (\mathbf{z} - \mathbf{z}_0), \tag{2.11}$$

where

$$r = \|\mathbf{z} - \mathbf{z}_0\|_2, \quad h(\alpha, r) = \frac{1}{\alpha + r},$$

and the parameters are $\mathbf{z}_0 \in \mathbb{R}^p$, $\alpha \in \mathbb{R}_+$, and $\beta \in \mathbb{R}$. This transformation contracts or expands the space radially around the reference point $\mathbf{z}_0$.

The determinant of the Jacobian of this transformation can be computed efficiently in linear time as

$$\left|\det \frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right| = (1 + \beta\, h(\alpha, r))^{p-1} \left(1 + \beta\, h(\alpha, r) - \beta\, \frac{r}{(\alpha + r)^2}\right). \tag{2.12}$$

Figure 2.5 demonstrates such a flow acting on the unit square with base probability $q_0(\mathbf{z}) \sim$ Uniform $[0, 1]^2$.

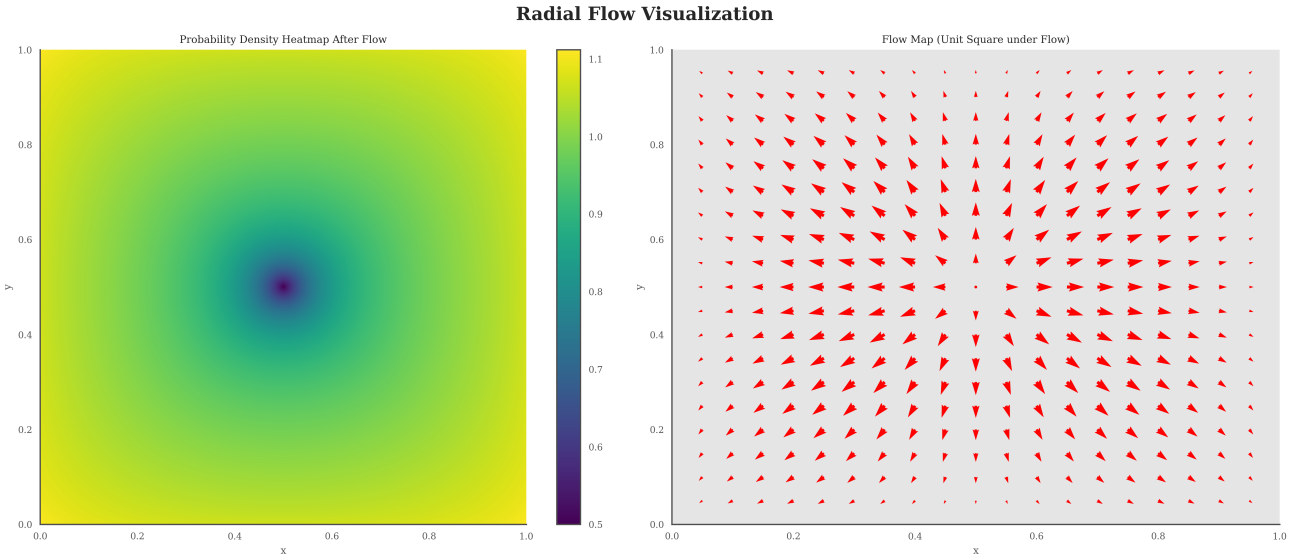


Figure 2.5: Probability density Heatmap (left) and Flow Map (right) showing the effect of a repulsive radial flow with $\alpha = 0.2, \beta = 0.3$. We see that probability is 'pushed' away from the center point and similarly points under the flow are transformed radially outwards.

Chapter 3

Methods

Motivated by the idea of replacing the acquisition function in the Bayesian Optimization loop, we turn to Normalizing Flows (NF) to help intuit what points to sample next. NFs are appealing because they allow us to flexibly shape probability distributions — attracting probability mass towards regions we believe to be promising while repelling it from areas we consider less favorable. This capability enables us to bias the model towards high-quality solutions and away from undesirable ones in a principled, probabilistic manner. We first present a high level overview of our model (FLOWBO) before detailing the specifics.

3.1 Outline

Figure 3.1 illustrates the overall structure of FLOWBO. Like most Bayesian optimization algorithms, the process begins by initializing the model with a set of data points, which may be chosen at random or selected strategically to probe specific regions of the search space. This initial sampling provides a foundation for distinguishing promising from unpromising areas and for gauging the relative potential of different locations.

Next, we introduce radial flows centered at each queried point: attractive flows in regions that appear promising, and repulsive flows where performance is poor. The strength and spatial extent of each flow are tunable, and these parameters are adaptively updated as additional data is acquired. To determine the appropriate "radius of influence", we fit a Gaussian Process to all data points at iteration n. A local approximation of the GP around each point is then used to characterize its behavior, allowing us to infer both the strength and influence of the associated flows in a principled manner.

Once the flows are established, we draw samples from the model, which is now calibrated to concentrate probability mass in favorable regions. Because the method is built on normalizing flows, it naturally preserves the ability to explore the broader search space, maintaining a balance

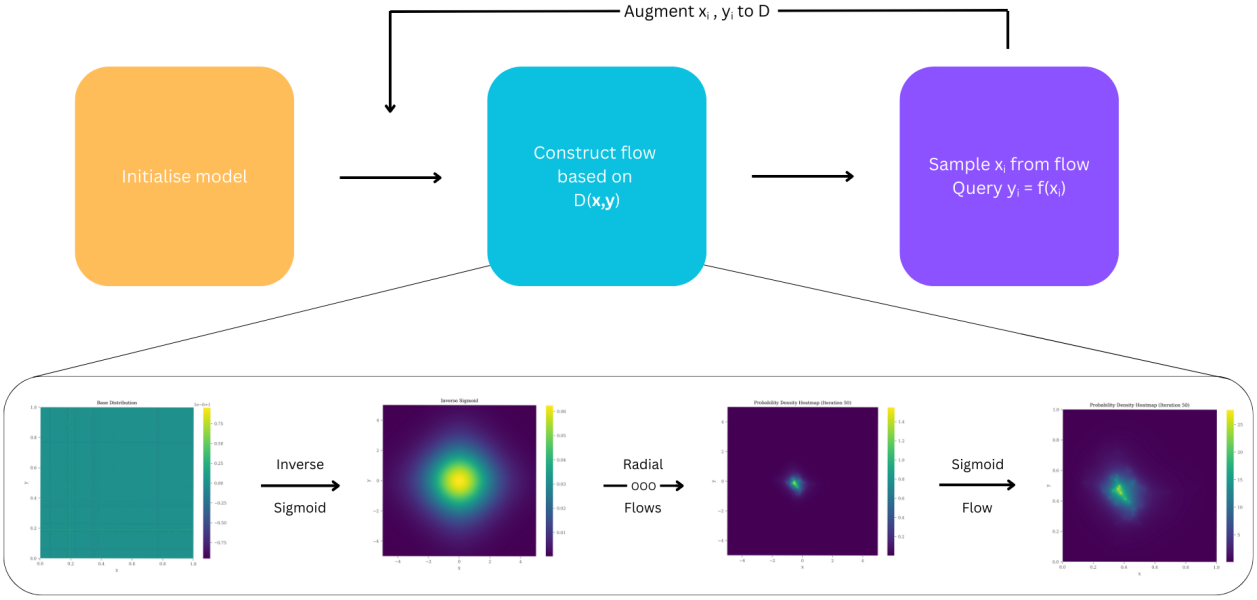


Figure 3.1: High level flow diagram of FLOWBO. Note how the model replaces the role of an acquisition function in the traditional BO loop. The heatmaps below demonstrate how our probability distribution is changed under each type of flow in the model: transforming a standard Uniform distribution to a targeted region determined adaptively by the data collected.

between exploitation and exploration. As new queries are added to the dataset, corresponding flows are introduced, and all flow parameters are dynamically updated to reflect the discovered landscape.

3.1.1 Unconstrained Latent Space

By observing 2.11 we see that the image of the normalizing flow does not respect any bounds constraining our problem; points from within the unit square may be mapped outside. From a probabilistic perspective this leads to a loss in probability within the problem space and also means our sampler will not necessarily provide plausible points. To address this issue we perform all radial flows in an unconstrained latent space which we map from and to using sigmoid and inverse sigmoid flows respectively. The second heatmap in Figure 3.1 shows the result of applying the inverse sigmoid flow to the Uniform distribution on the unit square. Refer to Appendix A for more details.

3.2 Choosing α and β

A radial flow around z_0 is described by parameters α and β, though these parameters do not directly correspond to specific physical features of the flow. Thus, to be able to control

the characteristics of each flow, depending on the nature of the point it is centered on, we re-parameterized the transformation.

From equation (2.10), we see that the inverse of the Jacobian determinant serves as a scaling factor for the base probability density. Therefore, if we want our radial flow to influence the space only up to r_{inf}, the determinant should approach 1 beyond this radius, ensuring the transformation preserves probability density outside the region of interest.

Similarly, we can control the scaling of the probability density at the center of the flow, denoted by c. Setting $c > 1$ increases the probability density around this point, emphasizing it if the point is compelling. Conversely, setting $c < 1$ reduces the probability density, effectively down-weighting the influence of less relevant points. Figure 3.2 shows how both parameters affect the nature of the transformation. Increasing r_{inf} means our flow changes the probability in a wider region around the centre point. We now discuss how to choose r_{inf} in a principled way using information from a global GP.

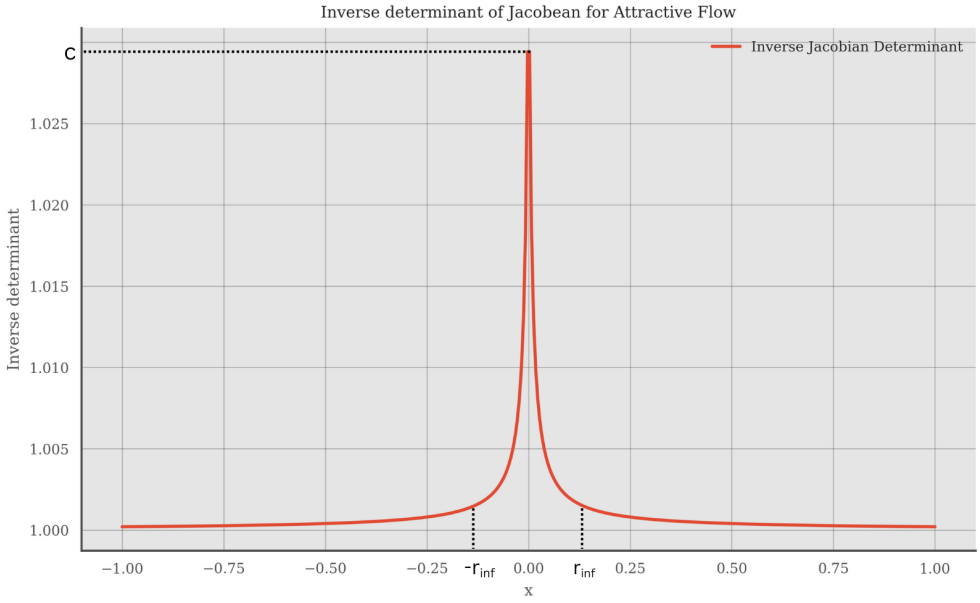


Figure 3.2: Plot of 1D Inverse Jacobian for an attractive radial flow. As this is attractive, the function is greater than 1, with c the maximum value at the centre of the flow. We see that for large distances, the determinant approaches 1 and the effect of the flow decreases.

3.2.1 Local GP Approximation

At every iteration of the model, we standardize the data points and then fit a GP to all points using the RBF kernel, in 2D this is:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{(x_1 - x_1')^2}{2\ell_1^2} - \frac{(x_2 - x_2')^2}{2\ell_2^2}\right) \tag{3.1}$$

where σ^2 is the output scale and ℓ_1, ℓ_2 are the length scales for the two dimensions (see Section

2.2.3). These parameters are updated with the addition of each new batch of data points.

To determine what r_{inf} should be, we first use an approximation of the global GP:

$$p(\mathbf{x} \mid (\mathbf{x}_i, y_i)_{i=1}^n \approx \prod_{i=1}^{n} p(\mathbf{x}|\mathbf{x}_i, y_i),$$

where we have assumed that each data point is independent. We then go one step further and say that around point i, $p(\mathbf{x} \mid \mathbf{x}_i, y_i)$, determines the majority of the behavior and is sufficient to model the local behaviour of the GP around point $(\mathbf{x}_i, y_i)$.

The term $p(\mathbf{x} \mid \mathbf{x}_i, y_i)$ can be interpreted as a *local Gaussian Process*, obtained by taking the parameters of the global GP, namely its mean function f_μ, output scale σ, and length scales ℓ_1, ℓ_2, and conditioning this GP on the specific observation $(\mathbf{x}_i, y_i)$. We need to make such an assumption as it allows for computationally feasible calculations of the quantities described in the following. Explicitly, we get:

$$f_i(\mathrm{x}) \sim N\left(f_\mu + \frac{\sigma^2 e^{-r_i^2}}{\sigma^2 + \sigma_e^2}(y_i - f_\mu), \sigma^2 - \frac{\sigma^4 e^{-2r_i^2}}{\sigma^2 + \sigma_e^2}\right), \tag{3.2}$$

where

$$r_i^2 = \frac{\sigma^2}{2}\left(\frac{(x - x_i)^2}{\ell_1^2} + \frac{(y - y_i)^2}{\ell_2^2}\right) \tag{3.3}$$

is the *length scale weighted distance* from the general point $\mathbf{x}$ to the observation $\mathbf{x}_i$.

In what follows, we focus on using FLOWBO to identify the global minimum. We thus define the annulus of influence for attracting flows, for a given K, to be

$$\mathcal{R}_{\text{inf}}^{\text{att}} \coloneqq \left\{ r_i(\mathbf{x}) \ : \ \mathbb{P}_i\left(f_i(\mathbf{x}) < f_{\min}\right) > K \right\} \subseteq [0, \infty), \tag{3.4}$$

i.e, the region in which our local GP believes there is a high enough probability of a point attaining a new global minimum. Likewise, the annulus of influence for a repelling flow, for a given K, is defined as:

$$\mathcal{R}_{\text{inf}}^{\text{rep}} \coloneqq \left\{ r_i(\mathbf{x}) \ : \ \mathbb{P}_i\left(f_i(\mathbf{x}) > f_{\max}\right) > K \right\} \subseteq [0, \infty),$$

the region in which our local GP believes there is high probability of attaining a new global maximum. The choice of K is explored in the next section.

Figure 3.3 shows a local Gaussian Process with it its posterior mean, confidence intervals (blue)

and $\mathcal{R}_{\text{inf}}^{\text{att}}$ (green). In the case presented we have two roots to the equality $\mathbb{P}_i(f_i(\mathbf{x}) < f_{\min}) = K$ and thus have a bounded interval for $\mathcal{R}_{\text{inf}}^{\text{att}}$. This allows for a clear interpretation: for r_i lower than the smallest root, there is not a high probability of achieving a new minimum however, the GP predicts that past this point, there is a small region in which it is likely there is one to be found. We now discuss the case when only root is found to the equality and such an interpretation is harder to extract.

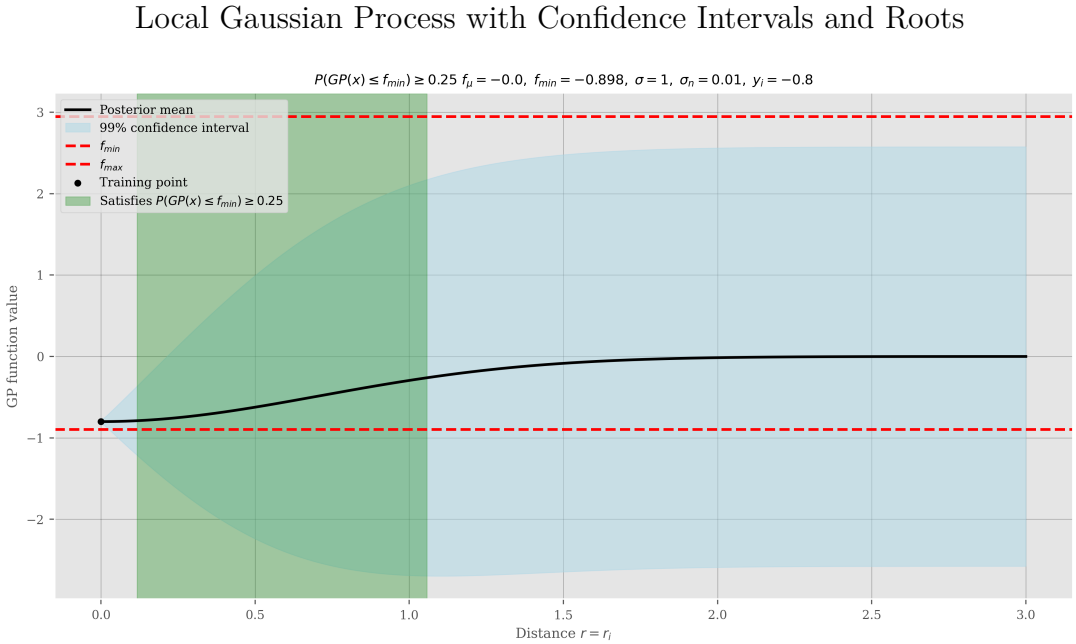


Figure 3.3: An example of a *local GP*. We see the posterior mean pass through the observed data point at (0, -0.8) and then return up to the mean of 0. The 99% confidence bounds show a slight dip, hinting the region where there is a higher probability of finding a global minima. The green band highlights those r_i which solve the inequality $\mathbb{P}_i(f_i(\mathbf{x}) < f_{\min}) > 0.25$.

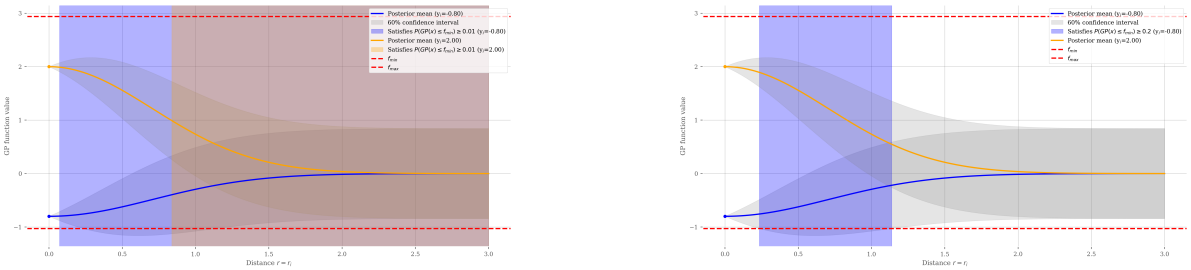
One root solutions

In the case that the equality $\mathbb{P}_i(f_i(\mathbf{x}) < f_{\min}) - K$ only has one root, we must explore the nature of $\mathcal{R}_{\text{inf}}^{\text{att}}$ more closely. Figure 3.4 illustrates the impact of varying K on two test points: one located near the current minimum at (0, -0.8), and another situated farther from the minimum at (0, 2). We see that with a small K, the inequality $\mathbb{P}_i(f_i(\mathbf{x}) < f_{\min}) > K$ is very relaxed and hence even poor performing points have a nonempty $\mathcal{R}_{\text{inf}}^{\text{att}}$ which is unbounded from above; see Figure 3.4a. Thus, to be able to differentiate between good and bad points, we should increase K, ensuring only truly promising points generate a radius of influence; see Figure 3.4b.

Adaptive K To address this issue, during initialization we evaluate the proportion of points that yield single roots, with the expectation that as K increases, the inequality constraint becomes stricter, allowing only the most promising points to have a non-empty $\mathcal{R}_{\text{inf}}^{\text{att}}$. We start from $K = 0.05$ and increment in steps of 0.05 until fewer than half of the queried points possess a non-empty $\mathcal{R}_{\text{inf}}^{\text{att}}$.

Figure 3.4: Comparing the effect of K on $\mathcal{R}_{\text{inf}}^{\text{att}}$

$K = 0.01$ $K = 0.2$



(a) Posterior means, 60% confidence intervals and $\mathcal{R}_{\text{inf}}^{\text{att}}$ for two test points, $(0, -0.8)$ and $(0, 2)$. With the low K and hence relaxed inequality constraint, both points have a non empty $\mathcal{R}_{\text{inf}}^{\text{att}}$.

(b) Posterior means, 60 % confidence intervals and $\mathcal{R}_{\text{inf}}^{\text{att}}$ for two test points, $(0, -0.8)$ and $(0, 2)$. With the higher K, the inequality imposes a stricter condition meaning only more desirable points have a non empty $\mathcal{R}_{\text{inf}}^{\text{att}}$.

Large σ in GP After accounting for the loose inequality constraint, we notice that many points may still yield a single root if the variance (output scale) of the local GP is large in comparison to the spread of obtained function values.

Figure 3.6 illustrates this phenomena. Due to the large σ the local GPs trained on poor points eventually assign enough probability below f_{min} to satisfy even the most strict inequality constraint, leading to the misleading interpretation that the given region is actually desirable. In this situation, despite K being large, we will still observe many data points producing a *single root*. To differentiate between the points, we therefore turn to how large this single root is: strong points are close to the global minima and thus the GP assigns probability below f_{min} much sooner than poorer points. We generate $\mathcal{R}_{\text{inf}}^{\text{att}}$ from the single roots r as $(0, u(r))$ where $u(r)$ is as shown in Figure 3.5. Implied in the transformation is the assumption that points seperated distances greater than the smallest length scale, $\min_i \ell_i$ are uncorrelated.

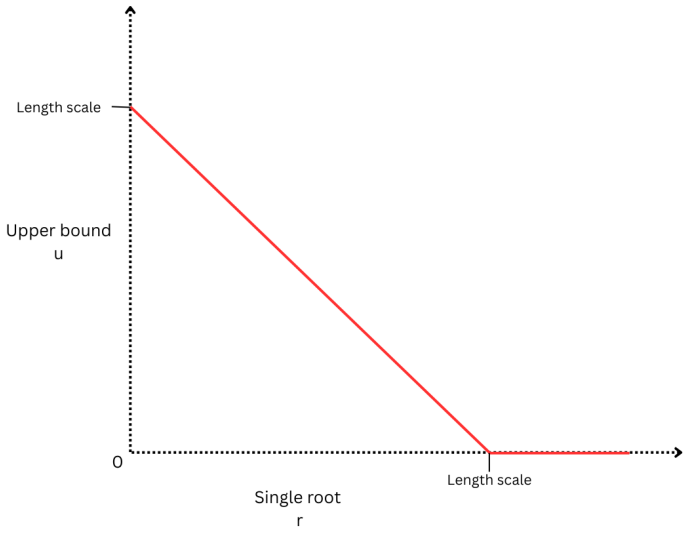


Figure 3.5: Plot of $u(r)$. Single roots larger than the GP's length scale demonstrate the point is weak and hence $\mathcal{R}_{\text{inf}}^{\text{att}}$ should be empty.

Local GP: $K = 0.4$, $\sigma = 4$

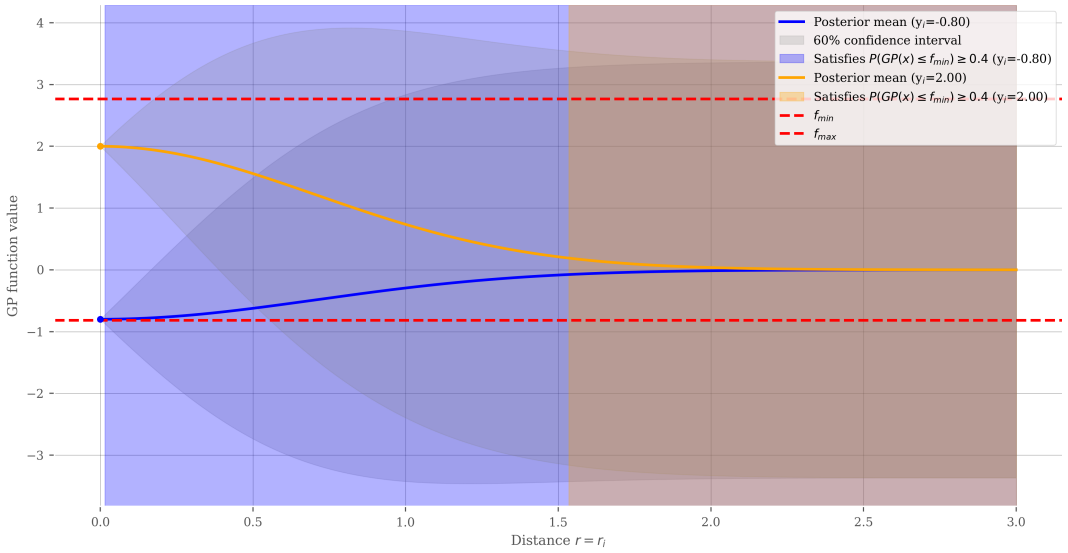


Figure 3.6: An example of a local GP with a large variance, $\sigma = 4$, compared to the spread of observed data points: $f_{\min} = -0.898$ and $f_{\max} = 2.95$. This results in local GPs trained on any data point eventually satisfying our inequality constraint. The key observation is that stronger points (in this case smaller) have a smaller lower bound of $\mathcal{R}_{\text{inf}}^{\text{att}}$

.

From $\mathcal{R}_{\text{inf}}^{\text{att}}$ to r_{inf}

Recall from equation 3.4 that $\mathcal{R}_{\text{inf}}^{\text{att}}$ consists of elements of r_i as defined in equation 3.3:

$$r_i^2 = \frac{\sigma^2}{2}\left(\frac{(x-x_i)^2}{\ell_1^2} + \frac{(y-y_i)^2}{\ell_2^2}\right)$$

These are length scale weighted distances and consequently level sets of r_i correspond to ellipses in the original space with semi-major and minor axes of lengths

$$l_i(r_i) = \sqrt{2}\ell_i \frac{r_i}{\sigma};$$

see Figure 3.7. Say without loss of generality that $l_2 < l_1$ as in the figure. We then take $r_{\text{inf}} = l_2\,(\sup \mathcal{R}_{\text{inf}}^{\text{att}})$.

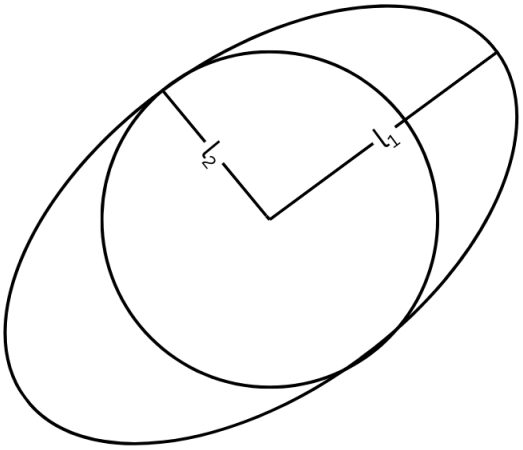


Figure 3.7: Level sets of r_i define ellipses with semi-major l_1, l_2 and minor axes as found to the left

Thresholding

Now each observed point in our model has a radial flow associated with it, with a specific r_{inf}. The spatial influence of nearby flows may interfere with each other, causing the local approximation to break and also leading to unintended relationships between close points. We thus test three different ways to restrict r_{inf}:

1. No thresholding applied
2. Threshold r_{inf} for point i by the distance to the closest point
3. Threshold by shortest length scale in the global GP

The comparison in performance of all three criterion is found in Results 4.1.

3.2.2 Center Values

To determine c, the probability density scaling at the center of the flow, we first assign each point as attractive or repulsive. A point is attractive if the corresponding $\mathcal{R}_{\text{inf}}^{\text{att}}$ is non-empty. If it is empty then a point is repulsive if $\mathcal{R}_{\text{inf}}^{\text{rep}}$ is non-empty. If both are empty then the point is near the mean of observed values and hence should have a minimal repulsion just to avoid resampling it. Once the points have been labeled the centers are then found by scaling the function values within each group to the desired ranges: 1.05 - 2 for attractive, 0.95 for insignificant and 0.5 - 0.95 for repulsive. Figure 3.8 illustrates the mapping.

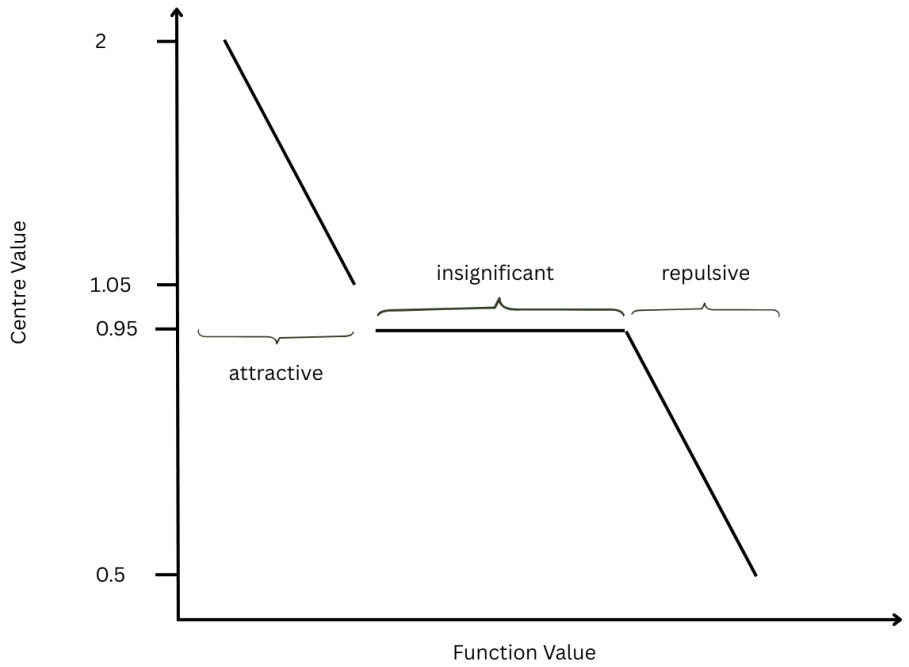


Figure 3.8: Scaling function for Center Values, c.

3.2.3 From c, r_{inf} to α, β

We defined c to be the value of the inverse determinant at the center of the flow. This gives us:

$$\left|\det \frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{r=0}^{-1} = c = \left(1 + \frac{\beta}{\alpha}\right)^d \tag{3.5}$$

$$\Rightarrow \beta = \alpha\left(c^{-d} - 1\right) = \alpha y. \tag{3.6}$$

We further defined r_{inf} as the radius at which the inverse determinant is equal to one, giving us:

$$\begin{aligned}\left|\det \frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{r=r_{\text{inf}}}^{-1} = 1 &= \left(1 + \frac{\beta}{\alpha + r_{\text{inf}}}\right)^{d-1}\left(1 + \frac{\beta}{\alpha + r_{\text{inf}}} - \frac{\beta r}{(\alpha + r_{inf})^2}\right) \\ &\approx (1+y)^d[1 - \frac{(d+1)yr_{inf}}{\alpha(1+c)}],\end{aligned}$$

with y the proportionality constant from equation (3.6).

By solving these two equations, we find expressions for α and β in terms of r_{inf} and c:

$$\alpha = \frac{(1+y)^{d-1} y r_{inf}(d+1)}{(1+y)^d - 1}$$

$$\beta = \alpha y.$$

3.2.4 Full Algorithm

We begin by initializing our model, sampling a fixed number of points either randomly or arranged in a grid. We then fit the global Gaussian Process using a Radial Basis Function (RBF) kernel. Following this, we infer local Gaussian Process approximations for each sampled point to better characterize local variability. We adjust the parameter K such that the number of single roots is less than half the total number of points (see paragraph 3.2.1). Once K is selected, we determine the radius of influence (section 3.2.1) for each point and apply thresholding accordingly (section 3.2.1). The center value for each flow is then calculated using a piecewise scaling function (section 3.2.2), allowing us to uniquely define each flow. After initialization, we iteratively sample new points, update or find parameters for all flows, add new flows as needed, and continue this process until the evaluation budget is exhausted.

Algorithm 2 FLOWBO: Bayesian Optimization Without Maximizing Acquisition

Require: Objective f to minimize, budget N, initial points I, threshold criterion $\mathcal{T} \in$ {None, Length scale, Closest Point}

Set base distribution $p \leftarrow \text{Uniform}([0,1]^2)$
Initialisation:
    Sample I points $\mathcal{X}_{\text{init}}$ (randomly or grid), standardise
$K \leftarrow 0.05$
while #single roots $< I/2$ **do**
    Fit GP to $\mathcal{X}_{\text{init}}$
    Find number of single roots
    $K \leftarrow K + 0.05$
end while
Queried points: $\mathcal{X} = \mathcal{X}_{\text{init}}$
SET ALPHAS AND BETAS
Add flows
while budget not exhausted **do**
    Sample x_{new} from generated flow
    Add x_{new} to queried points $\mathcal{X}$
    Standardise $\mathcal{X}$
    SET ALPHAS AND BETAS
    Add radial flow for x_{new} and update all radial flow parameters
end while

Algorithm 3 SET ALPHAS AND BETAS

Fit GP to all queried points $\mathcal{X}$
Use local GP approximation to compute r_{inf}
Categorise regions as attractive, repulsive, or insignificant
Compute center values by scaling function
Threshold radii based on selected criterion $\mathcal{T}$

Chapter 4

Results

Having outlined the methodological framework in the previous section, we now present an empirical evaluation of FLOWBO. We first investigate the robustness of the approach under different configurations of model parameters, assessing how these choices influence performance and stability. We then benchmark FLOWBO against standard state-of-the-art Bayesian optimization methods, highlighting both relative strengths and potential limitations.

To provide a rigorous benchmark, we evaluate FLOWBO on a suite of five widely used test functions: the Branin function (a two-dimensional multimodal benchmark with well-separated global minima), the Levy function (highly non-convex with many local minima), the Hartmann function (a standard higher-dimensional test with multiple interacting variables), the Ackley function (characterized by a nearly flat outer region with a narrow global basin), and the Three-Hump Camel function (a low-dimensional benchmark with a polynomial structure and multiple local minima). All the functions are rescaled to allow for optimization over the unit square.

4.1 Model Parameters

We begin by testing different combinations of the model parameters. Specifically, we vary the distribution of the initialisation points (labelled as IN in the figures below), using either a uniform grid or random sampling, and the choice of thresholding criterion (TH), which may be set to None, the shortest GP length scale, or the distance to the closest point (see 3.2.1). We also consider two strategies for the GP output scale σ: either keeping it fixed at 1, or allowing it to be adaptively updated after each training step. This is done despite the standardization of the data points performed before the fitting. For each different possible combination of parameters, we perform 30 runs of the model, each running for 40 iterations with an additional 9 points used for initialization. Figures 4.1 to 4.4 display the average minima found across the runs along with standard error bars.

From the experiments we observe some key phenomena. First, for the Ackley and Three Hump Camel functions we see optimal performance with minimal standard error. However, on closer inspection, this is purely because one of the initial grid points is already located at the global minima. For the Branin and Levy functions, this is not the case and we infer:

- The effect of fixing the GP output scale varies across functions. For Branin, 5 out of 6 combinations of thresholds and initialization strategies performed better with a fixed σ, whereas for other functions the effect was less consistent.
- The best-performing configurations consistently included random initialization.
- Removing thresholding altogether leads to much larger standard errors, as expected, since attractive or repulsive points can have a strong influence early in the optimization.
- There is little difference in performance between length scale thresholding and closest-point thresholding.

Based on these findings, we compare FLOWBO with standard BO under random initialization and length scale thresholding. Since the impact of updating σ is inconclusive, we evaluate performance both with and without this update and benchmark against standard BO.

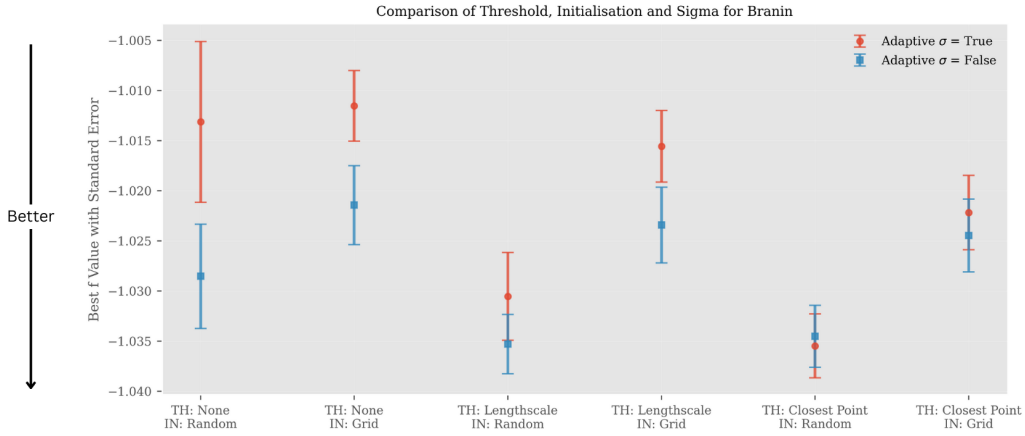


Figure 4.1: Scatter plot with standard errors on the Branin function, comparing different combinations of model parameters. The best-performing models use random initialization, while adaptive σ generally leads to poorer performance.

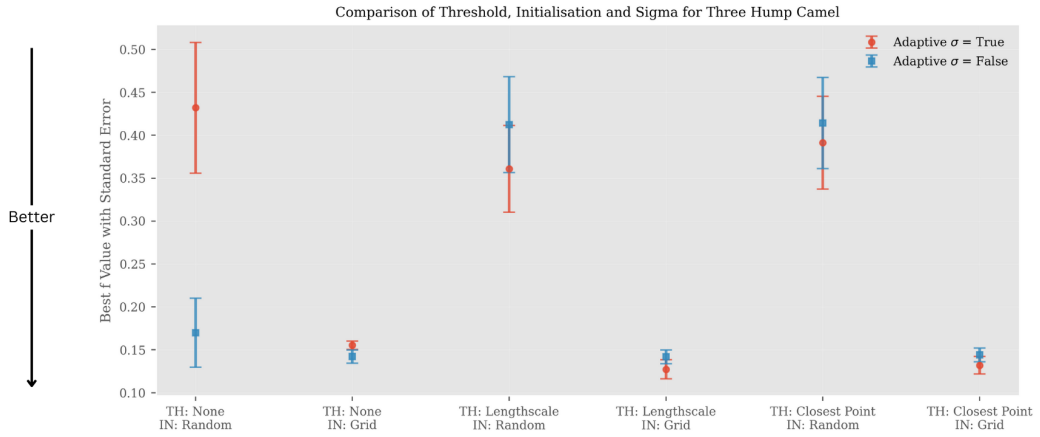


Figure 4.2: Scatter plot with standard errors on the Three-Hump Camel function, comparing different combinations of model parameters. Grid-based methods show optimal performance as one initial point coincides with the global minimum. With random initialization, adaptive σ has a mixed effect, and the combination of no thresholding with adaptive σ yields large standard errors.

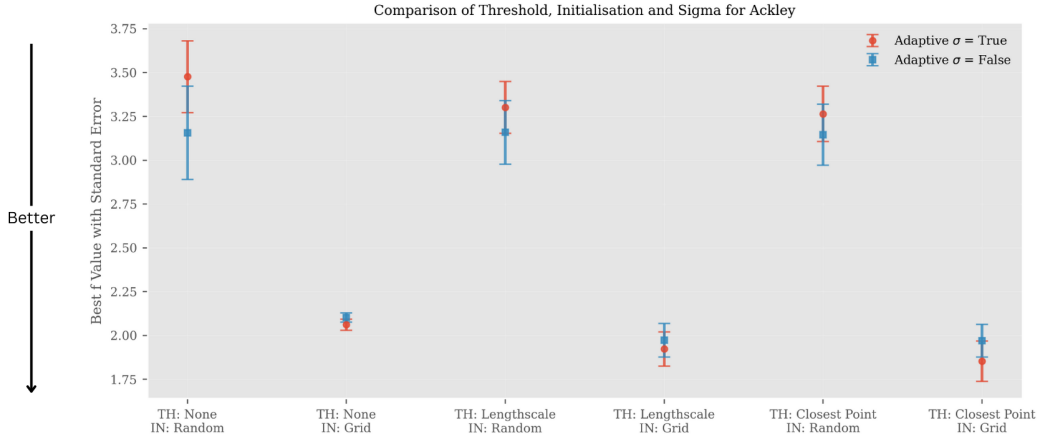


Figure 4.3: Scatter plot with standard errors on the Ackley function, comparing different combinations of model parameters. Grid-based methods again perform best as an initial point coincides with the global minimum. The effect of adaptive σ is less clear in this case.

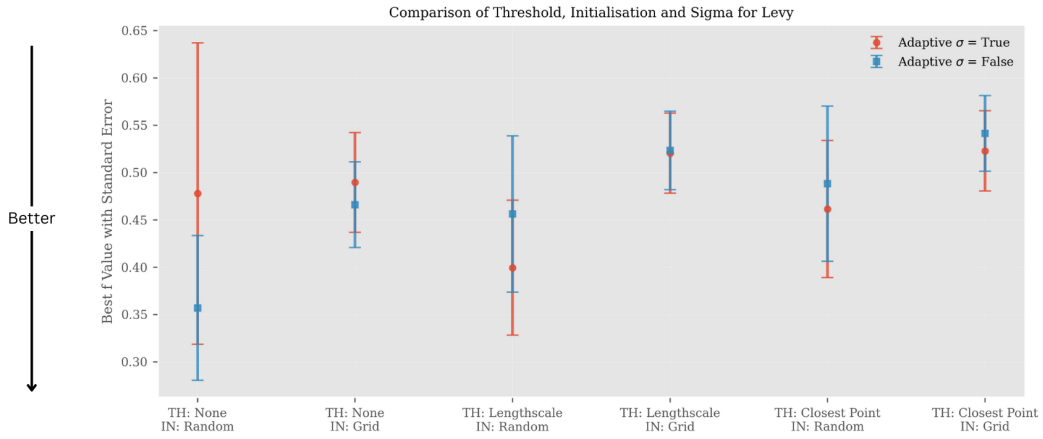


Figure 4.4: Scatter plot with standard errors on the Levy function, comparing different combinations of model parameters. Surprisingly, the no-threshold setting achieves the best mean performance. Random initialization has larger standard errors than grid-based methods but generally performs better. The effect of adaptive σ remains inconclusive.

4.2 Comparing to standard BO

We now compare FLOWBO against a traditional Bayesian optimization baseline in terms of mean performance, variability (standard error), and computational speed. In the baseline approach, at each iteration a Gaussian Process model with a Matérn 5/2 kernel is fitted to the observed data by maximizing the exact marginal log likelihood. The acquisition function used is the Log Expected Improvement, which provides a numerically stable alternative to the classical Expected Improvement. To maximize this acquisition function and select the next evaluation point, we use the L-BFGS-B optimizer with multiple restarts. Specifically, 20 candidate points are initially sampled uniformly within the search bounds, and the best 5 of these candidates serve as starting points.

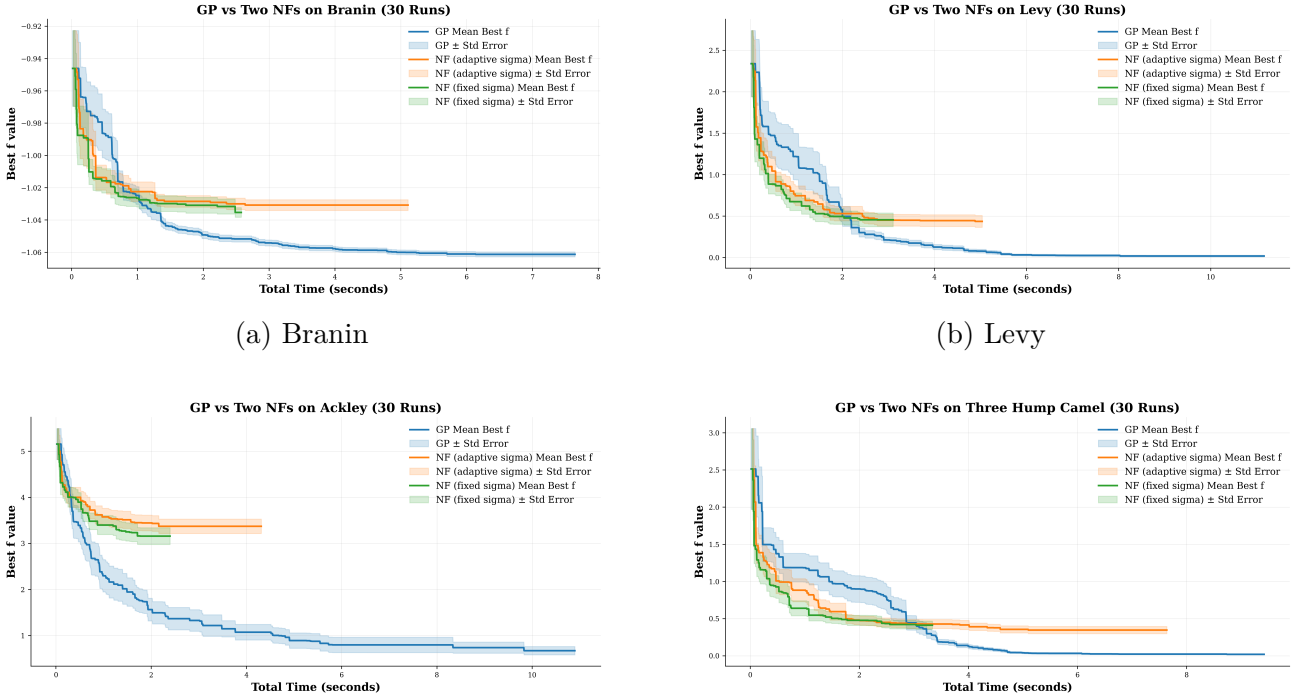
Figure 4.5 presents the results of experiments on the Branin, Levy, Ackley, and Three-Hump Camel functions. To ensure fairness, both methods are initialized from the same random starting points. We use an observation noise of 0.01, initialize with 9 points, and run each model for an additional 40 iterations. The results over 30 runs reveal the following:

- FLOWBO performs well and even outperforms standard BO during the early stages.
- FLOWBO eventually stops improving and all further points sampled make minimal difference. We verified this was the case by running for more than 80 samples and there was little improvement.
- FLOWBO is consistently quicker than the GP. Since both models fit a GP at each step, the speed increase is from not needing to maximise the acquisition function.
- FLOWBO struggles heavily to minimise the Ackley function.
- By fixing σ, the model runs much quicker and performs similarly, if not better, than those with adaptive σ.

4.2.1 Higher Dimension Optimization

Since our central goal was to avoid reliance on an acquisition function, which becomes increasingly difficult to optimize in higher dimensions, we evaluated FLOWBO on higher dimensional problems.

Figures 4.6 and 4.7 show the performance of FLOWBO on the 12 dimensional Levy function and the 6 dimensional Hartmann function. Although FLOWBO achieves lower optimization performance than standard BO, it is substantially faster, running more than three times quicker on Levy and twice as fast on Hartmann.



(a) Branin

(b) Levy

(c) Ackley

(d) Three-Hump Camel

Figure 4.5: Results from numerical experiments comparing FLOW and standard BO. We see FLOWBO performs the same number of iterations substantially quicker, but eventually suffers from over-exploitation and fails to improve thereafter.

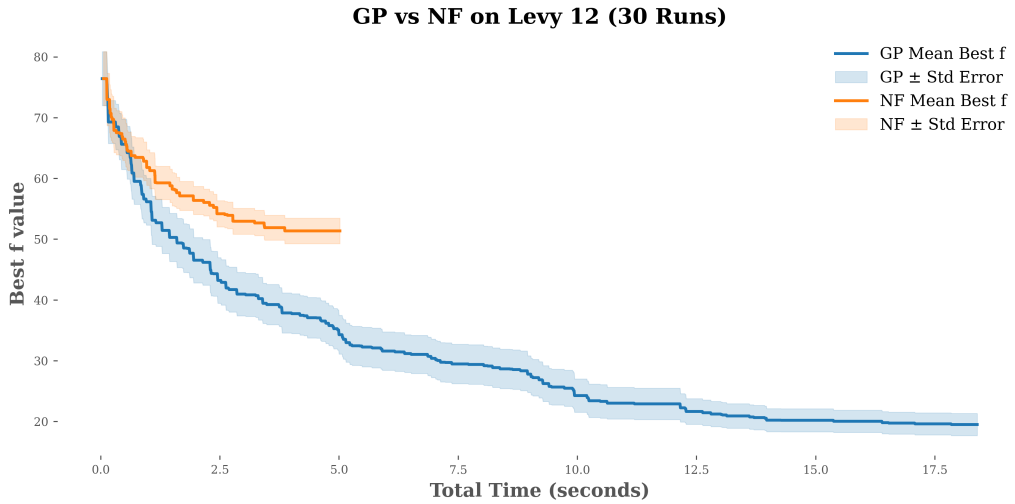


Figure 4.6: Results from experiments on the 12 dimensional Levy function. We note the 3x speed increase with FLOWBO

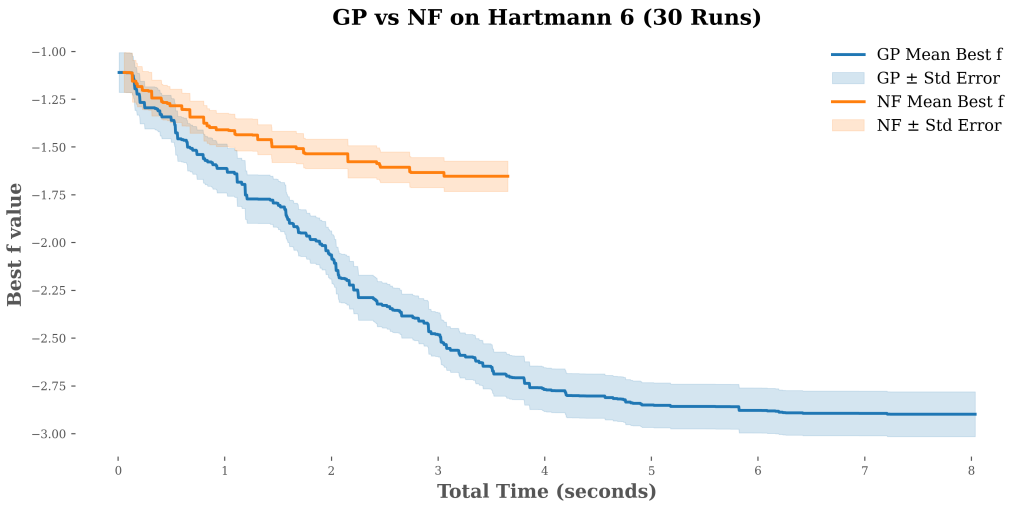


Figure 4.7: Results from experiments on the 6 dimensional Hartmann function. We note the 2x speed increase with FLOWBO

Chapter 5

Discussion

5.1 Speed Increase

From Figures 4.5 and 4.6, we found that FLOWBO consistently outperforms standard BO in terms of speed, with the advantage becoming especially pronounced in higher dimensions. The very motivation for developing FLOWBO was the difficulty of maximizing the acquisition function in standard BO. Since both FLOWBO and the baseline BO fit a GP to all points at each step, the difference in runtime arises entirely from FLOWBO's ability to bypass the acquisition function. As shown in Figure 5.1, the runtime of FLOWBO remains nearly constant as dimensionality increases, which presents a significant practical benefit.

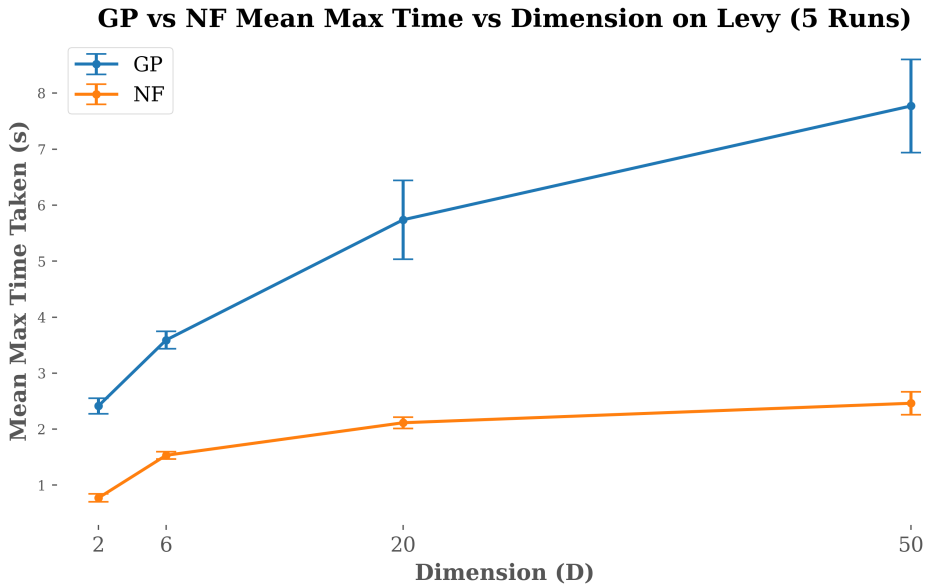


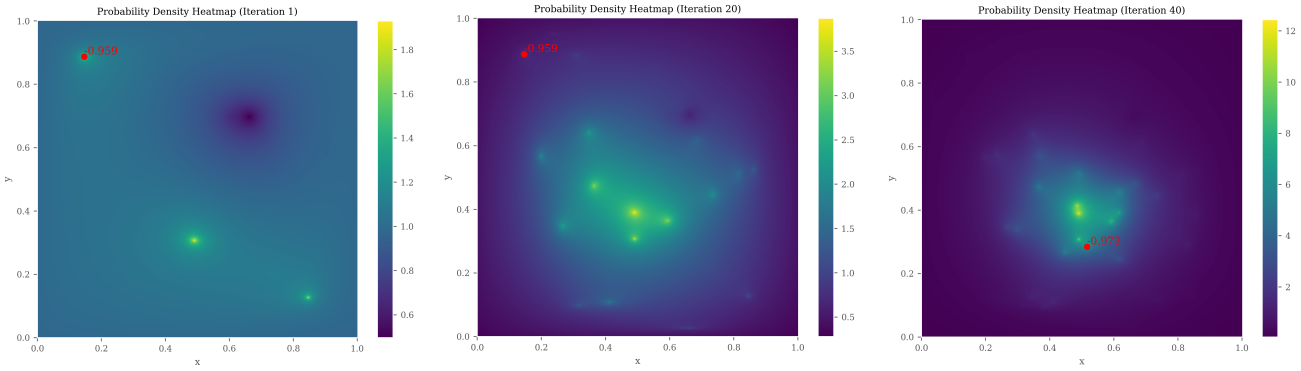
Figure 5.1: FLOWBO's speed advantage over standard BO becomes increasingly dominant as the dimension of the problem increases.

5.2 Scale Problem

From Figure 4.5, we observe that all of the FLOWBO models exhibit rapid progress during the early stages of optimization. This reflects the model's ability to quickly identify promising regions of the search space and concentrate probability mass there, leading to strong initial gains. However, this same concentration eventually becomes a limitation: as probability collapses into narrow regions, exploration is reduced and the model struggles to escape local modes. Consequently, the FLOWBO models stagnate and stop discovering further improvements. In contrast, the GP-based approach maintains a steadier balance between exploration and exploitation, allowing it to continue making incremental gains over time. As a result, while FLOWBO demonstrates superior early performance, it ultimately lags behind the GP baseline in the long run.

5.2.1 σ Too High

We first explore the problem of scale with the Branin function which displayed over exploitation and no exploration after approximitately 1.5s of run time (Figure 4.5a). To explore this further, we look at a single run of the model and see how the probability landscape changes through iterations 1, 20 and 40.



(a) Iteration 1 (b) Iteration 20 (c) Iteration 40

Figure 5.2: Evolution of the probability density as the flow updates with additional points. As expected, the region of interest narrows over time. However, the model becomes overly focused on the previously sampled region, rather than exploring the dip of the Branin function. Note how the scale changes across the figures representing the model becoming more confident. We plot the current best minimum found in red to demonstrate this is largely ignored by the model.

Figure 5.2 reveals that the model becomes increasingly focused in a particular region, as seen from the heatmaps' scales. While this in principle is a behavior we wish to see, the area being honed in on is constructed in the wrong way. We would wish for the model to observe a global minimum and explore the surrounding area in which it suspects will provide an improvement. Instead, what is happening here is that the model concentrates on where most of the previous

samples are located. In this case, the points cluster along the edge of one of Branin's valleys, rather than exploring within the valley itself as the GP does.

The reason for this is a problem of *scale*, σ, given by the fitted local GP. Due to the shallow valleys present in the Branin function, many of the sampled points give similar function values to the global minima. While this should be accounted for in the local GP, we instead find that an incredibly high σ is fitted when using an adaptive output scale. Even fixing $\sigma = 1$ is too large for the model to discern truly strong points.

This means that the model believes every point is promising (see Figure 3.6) and further many of the centres are scaled all quite close to 2. This is because we scale the attraction centres based on the function values of those points deemed attractive (see Section 3.2.2). However, this now also includes the worst point sampled by the model, meaning there are a large number of points with a higher scaling. This along with the probability density scaling interpretation of the flows (Eq: 2.10) explains the large probability density around the sampled points.

Similar behavior is observed when the model runs on the Levy and Three Hump Camel functions. Both of these have a very shallow valley bordered by sharp edges. The model identifies the valley floor, however, is unable to appropriately discern strong points within it. Figure 5.3 allows us to visualize this behavior by plotting the percentage of flows which are deemed attractive in each iteration. We see that for these three functions, the percentages are notably high.

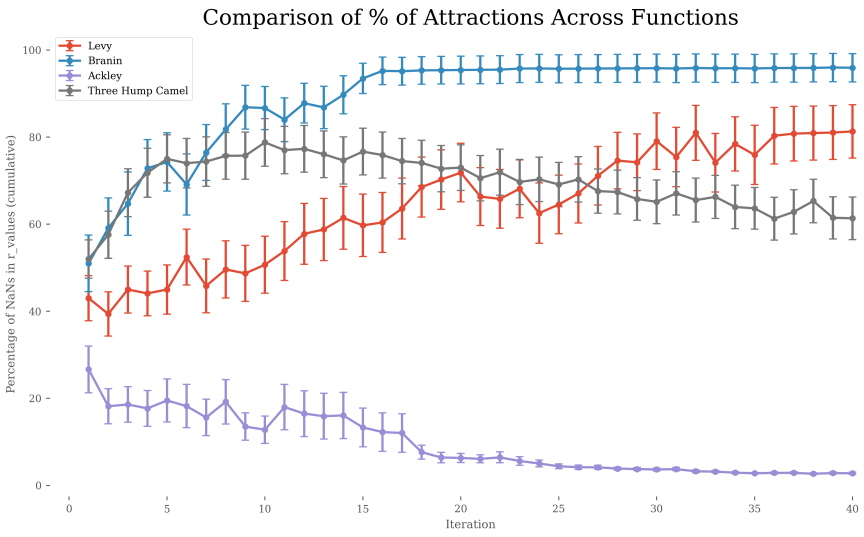


Figure 5.3: Scatter plots with standard errors showing the % of attractive flows present at each iteration. We see for those functions with minima in shallow regions, our model struggles to selectively assign attractive flows.

5.2.2 σ Too Low

While the Branin, Levy, and Three-Hump Camel functions suffer from the local GP approximation assigning too large an output scale, the Ackley function exhibits the opposite issue. The Ackley

function contains many small oscillations and a single sharp drop, producing a highly uneven landscape. This highly multi-modal landscape is unrepresentative of the functions we aim to minimize, but it highlights a weakness of our model. Because of the rapid changes in the function, the model tends to sample values far from the current minimum, leading it to treat most points as uninformative. We may count the number of points treated as uninformative by seeing how many roots are found when finding $\mathcal{R}_{\text{inf}}^{\text{rep}}$ or $\mathcal{R}_{\text{inf}}^{\text{att}}$. If none are found the solver returns a NaN. Since our approach relies on local approximations, it performs poorly on Ackley. As shown in Figure 5.4, the percentage of NaNs approaches 100% coinciding with our model stagnating.

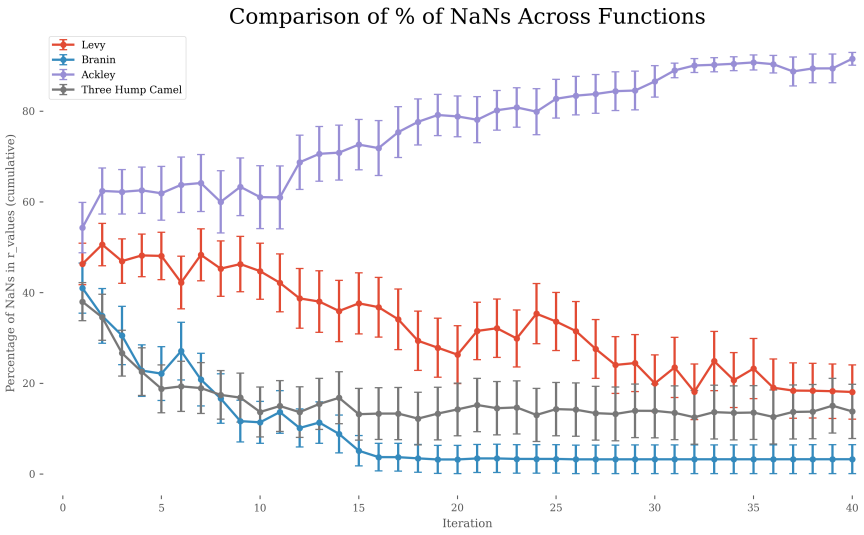


Figure 5.4: Scatter plots with standard errors showing the % of insignificant (NaN) flows present at each iteration.We note in the case of Ackley, this approaches 100% conveying the model is unable to identify any regions as promising.

Chapter 6

Conclusion

In this dissertation, we have introduced Bayesian Optimization as a principled strategy for optimizing objective functions that are expensive to evaluate. At the heart of Bayesian optimization lies the interplay between the surrogate model, which provides a probabilistic approximation of the objective, and the acquisition function, which guides the selection of new sampling points. We focused in particular on Gaussian Processes; a flexible generalization of the Multivariate Normal Distribution that preserves key properties such as marginalization and conditioning. Special attention was paid to the influence of the prior mean, and especially the importance of carefully defining the model's kernel, which encodes assumptions about the smoothness and structure of the target function.

We also examined the construction and use of acquisition functions that mediate the trade-off between exploration and exploitation, determining how new evaluations are chosen in the optimization loop. However, BO faces significant practical challenges, notably the difficulty of maximizing acquisition functions in high-dimensional spaces—a scenario frequently encountered when tuning hyperparameters in complex machine learning models.

To overcome this issue, we developed a novel alternative to conventional acquisition functions by utilizing normalizing flows to guide the selection of future sampling locations. To better understand the properties of these flows, we first re-parameterized from the original variables (α, β) to more interpretable quantities: the center value, c, and the radius of influence, r_{inf}. To determine the spatial extent of each flow, we approximated the global Gaussian Process surrogate with a collection of *local Gaussian Processes*, each characterizing the behavior around a specific point. The centre value c was then determined using a piecewise linear scaling function, making it directly responsive to the function value at each location.

After experimentation, we found that our method was faster than standard Bayesian optimization, and the time required per iteration remained near constant with the dimensionality of the problem. However, a notable limitation emerged: our model exhibited a tendency to over-exploit, focusing

too much on regions it has previously sampled in. Further investigation revealed that this behavior was due to the local Gaussian Process surrogates lacking the ability to reliably distinguish poor candidate points from genuinely promising ones.

6.1 Future Work

Future work should focus on addressing the limitations uncovered in this dissertation. A primary goal is to improve the model's ability to accurately differentiate between promising and poor points. This may involve reworking the local Gaussian Process model by refining the parameter K, which governs the strictness of classifying points as good or bad (see 3.2.1). In the current version of the model, this is only selected during initialization, however it may be beneficial to adaptively update this in hopes to avoid the phenomena found in the Discussion 5.2.1, where all points were found to be attractive. We currently choose K to limit the number of single roots, however, this may be changed to control the number of insignificant points (no $\mathcal{R}_{\text{inf}}^{\text{rep}}$ or $\mathcal{R}_{\text{inf}}^{\text{att}}$).

Additionally, we may want to reassess the use of the output scale of the fitted GP, as experiments, for instance with the Branin function, revealed that very high output scale values cause local models to degrade even over relatively small distances. A potential solution could be to fit many local Gaussian Processes, as done in methods like TuRBO [5]. This approach would be especially beneficial in cases where large regions exhibit shallow gradients, and the global GP's length scale does not adequately represent local variability.

Further developments could explore more flexible geometric transformations for the flows, such as utilizing ellipsoids shaped according to the length scales learned by the GP, instead of simpler radial forms. To mitigate over-exploitation, introducing an annulus of repulsion around sampled points could help prevent clustering by discouraging samples from being too close to each other. Early investigations into this approach indicated challenges in selecting suitable parameters, as first-order approximations tend to break down under these conditions.

Finally, splines present an exciting opportunity to expand the modeling paradigm by providing flexible, principled deformations of the probability space between sampled points. This approach could allow for smoother and more adaptive probability transformations, ultimately improving the exploration-exploitation balance in Bayesian optimization.

AI Statement

AI was used to help stylize plots in this dissertation.

Appendix A

Loss of Probability

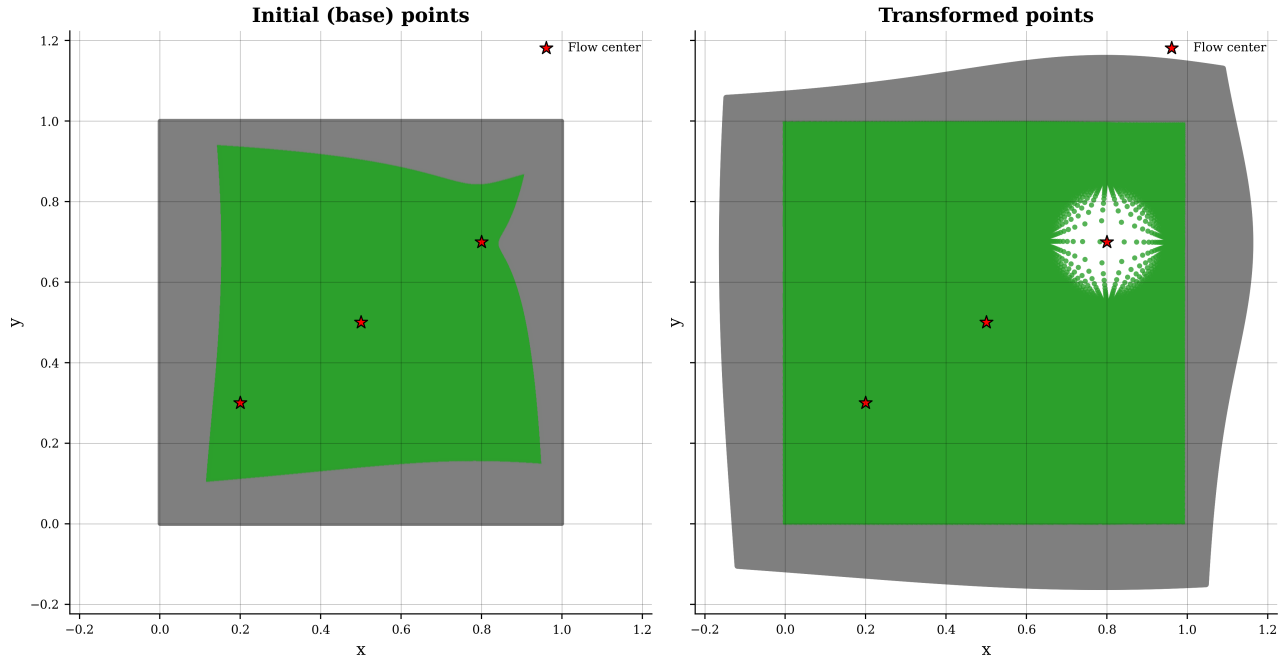


Figure A.1: The green shape on the left represents where valid points may be sampled from. With more flows this area decreases, motivating the need for a latent space.

Figure A.1 demonstrates the necessity to first map our unit square into an unconstrained latent space (Sec: 3.1.1). Our radial flows are applied to base space, in which the unit square is given by the grey box. The centres of our flows are marked by the red stars. The gray unit square is then transformed under this flow, the image of which is given by the gray region in the right-hand figure. We note the slight bulges from the stronger flow. We mark the unit square in the transformed space with the green box and its pre image is also marked by green in the base space. The green in our base space thus represents the set of points which we may begin our flow from to generate plausible sampled points. As the number of flows increases, this region decreases in size, representing how the radial flows do not conserve probability. This problem is avoided by mapping to $\mathbb{R}^2$.

Bibliography

[1] Rika Antonova, Akshara Rai, Tianyu Li, and Danica Kragic. Bayesian optimization in variational latent spaces with dynamic compression. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 456–465. PMLR, 2020.

[2] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space, 2016. URL https://arxiv.org/abs/1511.06349.

[3] Francesco Paolo Casale, Adrian Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian process prior variational autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[4] George De Ath, Jonathan E. Fieldsend, and Richard M. Everson. What do you mean? the role of the mean function in bayesian optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO 2020)*, pages 1623–1631. ACM, 2020.

[5] David Eriksson, Michael Pearce, Jacob R Gardner, Ryan Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization, 2020. URL https://arxiv.org/abs/1910.01739.

[6] Peter I. Frazier. Bayesian optimization. In *Recent Advances in Optimization and Modeling of Contemporary Problems*, pages 255–278. INFORMS, 2018.

[7] Peter I. Frazier. A tutorial on bayesian optimization, 2018. URL https://arxiv.org/abs/1807.02811.

[8] Milton Friedman and Leonard J. Savage. Planning experiments seeking maxima. In *Selected Techniques of Statistical Analysis for Scientific and Industrial Research, and Production and Management Engineering*. 1947.

[9] Joseph Diez Gergonne. Application de la méthode des moindres quarrés à l'interpolation des suites. *Annales de Mathématiques Pures et Appliquées*, 6:242–252, 1815.

[10] Samuel Gershman, Matt Hoffman, and David Blei. Nonparametric variational inference, 2012. URL https://arxiv.org/abs/1206.4665.

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

[12] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks, 2014. URL https://arxiv.org/abs/1310.8499.

[13] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 11(2): 577–586, Nov 2019. doi: 10.1039/c9sc04026a.

[14] Tommi Jaakkola and Michael Jordan. Improving the mean field approximation via the use of mixture distributions. 08 1997. doi: 10.1007/978-94-011-5014-9_6.

[15] Noémie Jaquier, Leonel Rozo, Sylvain Calinon, and Mathias Bürger. Bayesian optimization meets riemannian manifolds in robot learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 233–246. PMLR, 2019.

[16] Donald Jones, Matthias Schonlau, and William Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 12 1998. doi: 10.1023/A:1008306431147.

[17] Michael Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 01 1999. doi: 10.1023/A:1007665907178.

[18] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric Xing. Neural architecture search with bayesian optimisation and optimal transport, 2018. arXiv preprint.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015. arXiv:1412.6980.

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR)*, 2014. arXiv:1312.6114.

[21] Jeffrey Larson, Matt Menickelly, and Stefan M. Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, May 2019. ISSN 1474-0508. doi: 10.1017/s0962492919000060. URL http://dx.doi.org/10.1017/S0962492919000060.

[22] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. Chapter 28.

[23] Jonas Mockus. *The Bayesian Approach to Local Optimization*. Springer, 1989.

[24] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for bayesian

optimization in embedded subspaces. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 4752–4761. PMLR, 2019.

[25] ChangYong Oh, Efstratios Gavves, and Max Welling. BOCK: Bayesian optimization with cylindrical kernels. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3868–3877. PMLR, 2018.

[26] Leonard Papenmeier, Matthias Poloczek, and Luigi Nardi. Understanding high-dimensional bayesian optimization, 2025. URL https://arxiv.org/abs/2502.09198.

[27] C. S. Peirce. Note on the theory of the economy of research. In *Report of the Superintendent of the United States Coast Survey Showing the Progress of the Work for the Fiscal Year Ending with June, 1876*, pages 197–201. Government Printing Office, 1876.

[28] Mercy Prasanna Ranjit, Gopinath Ganapathy, Kalaivani Sridhar, and Vikram Arumugham. Efficient deep learning hyperparameter tuning using cloud infrastructure: Intelligent distributed hyperparameter tuning with bayesian optimization in the cloud. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 520–522. IEEE, 2019.

[29] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL http://www.gaussianprocess.org/gpml/.

[30] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016. URL https://arxiv.org/abs/1505.05770.

[31] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1278–1286. PMLR, 2014.

[32] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. URL https://arxiv.org/abs/1606.03498.

[33] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2015.

[34] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012. URL https://arxiv.org/abs/1206.2944.

[35] Michael L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, New York, 2012. ISBN 9781461214946.

[36] Matteo Turchetta, Andreas Krause, and Sebastian Trimpe. Robust model-free reinforcement learning with multi-objective bayesian optimization. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10702–10708. IEEE, 2020.

[37] Richard Turner, M. Sahani, D. Barber, Ali Cemgil, and S. Chiappa. Two problems with variational expectation maximisation for time-series models. *Bayesian Time Series Models*, pages 109–130, 01 2011.

[38] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. Appendix A.

[39] Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[40] Abraham Wald. *Sequential Analysis*. John Wiley & Sons, 1947.

[41] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent advances in bayesian optimization, 2022. URL https://arxiv.org/abs/2206.03301.

[42] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[43] Jian Wu, Saul Toscano-Palmerin, Peter I. Frazier, and Andrew Gordon Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 788–798. PMLR, 2020.

[44] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23:550–560, 1997.

[45] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer-Verlag, Berlin, Heidelberg, 6th edition, 2013.