Institute for Data Science in Mechanical Engineering

RWTH AACHEN UNIVERSITY

The present work was submitted to the Institute for Data Science in Mechanical Engineering.
Diese Arbeit wurde vorgelegt am Institut für Data Science im Maschinenbau.

Reducing the Influence of Model Mismatch in Bayesian Optimization Through Targeted Noise Injection

Reduktion des Einflusses von Modellabweichungen in der Bayes'schen Optimierung durch gezielte Zugabe von Messunsicherheiten

Bachelor Thesis
Bachelorarbeit

Presented by / Vorgelegt von

Julian Thorsten Winking
Matr.Nr.: 430952

Supervised by / Betreut von — Paul Brunzema, M.Sc.
(Institute for Data Science in Mechanical Engineering)

1st Examiner / 1. Prüfer — Univ.-Prof. Dr. sc. Sebastian Trimpe
(Institute for Data Science in Mechanical Engineering)

2nd Examiner / 2. Prüfer — Dr. rer. nat. Friedrich Solowjow
(Institute for Data Science in Mechanical Engineering)

Aachen, July 28, 2025

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne die Benutzung anderer als der angegebenen Hilfsmittel selbstständig verfasst habe; die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe des Literaturzitats gekennzeichnet.

Aachen, den 28. Juli 2025

Abstract

Bayesian optimization (BO) is a powerful and sample-efficient framework widely used to optimize expensive-to-evaluate black-box functions. It is applicable across various scientific and engineering domains, including robotics and controls. The performance of BO relies on the fidelity of the surrogate model. A critical challenge in practical BO applications is the inherent model mismatch between the assumed Gaussian process (GP) surrogate model and the true objective function, particularly when early, low-information observations dominate the GP's hyperparameter learning. This can result in poor surrogate accuracy near the global optimum, which hinders optimization performance.

To address this challenge, this thesis explores how observation noise can be adaptively adjusted to reduce the impact of misleading observations. We introduce targeted noise injection, a data-centric strategy that treats observation noise as a tunable modeling variable and selectively down-weights individual data points without modifying the GP architecture. This approach centers on a novel multiplicative Gaussian likelihood that enables per-point noise scaling.

In our evaluation, we consider both optimization strategies and structural extensions to identify suitable noise vectors used in targeted noise injection. First, we explore gradient-based methods as well as heuristic search strategies, including iterated local search and beam search, to optimize the noise configuration. Second, we introduce a trust region-guided variant that reduces the influence of observations outside the current region of interest. This enables the model to more effectively allocate capacity to the most relevant regions of the input space. Empirical results on synthetic optimization benchmarks show that targeted noise injection improves optimization performance compared to standard output transformations. These findings indicate that targeted noise injection is a practical and flexible mechanism for improving the reliability of BO in the presence of model mismatch.

Kurzzusammenfassung

Bayes'sche Optimierung (BO) ist ein leistungsstarkes und stichprobeneffizientes Verfahren zur Optimierung kostenaufwändig zu bewertender Black-Box-Funktionen. Sie findet breite Anwendung in verschiedenen wissenschaftlichen und ingenieurtechnischen Bereichen, darunter Robotik und Regelungstechnik. Die Leistungsfähigkeit von BO hängt maßgeblich von der Genauigkeit des verwendeten Surrogatmodells ab. Eine zentrale Herausforderung in praktischen BO-Anwendungen besteht in der inhärenten Modellabweichung zwischen dem angenommenen Gaußprozess-Modell (GP) und der tatsächlichen Zielfunktion, insbesondere wenn frühe, informationsarme Beobachtungen die Hyperparameterschätzung dominieren. Dies kann zu einer unzureichenden Modellgüte in der Nähe des globalen Optimums führen und damit die Optimierungsleistung beeinträchtigen.

Zur Bewältigung dieser Herausforderung untersucht diese Arbeit, wie Messunsicherheiten individuell angepasst werden können, um den Einfluss irreführender Beobachtungen zu verringern. Hierzu wird die gezielte Zugabe von Messunsicherheiten als datenorientierte Erweiterung eingeführt. Wir behandeln das Beobachtungsrauschen als justierbare Modellierungsvariable, mit der sich einzelne Datenpunkte selektiv abwerten lassen, ohne die Architektur des GP zu verändern. Zentral in diesem Ansatz ist eine neu entwickelte, multiplikative gaußsche Likelihood, die eine skalierbare Rauschmodellierung auf Punktbasis ermöglicht.

Zur Untersuchung des Ansatzes werden verschiedene Optimierungsstrategien und strukturelle Erweiterungen betrachtet, um geeignete Rauschvektoren für die gezielte Zugabe von Messunsicherheiten zu identifizieren. Zunächst werden Verfahren auf Gradientenbasis sowie heuristische Suchstrategien, darunter iterierte lokale Suche und Beam Search betrachtet. Anschließend wird eine trust-region-geführte Variante vorgestellt, die den Einfluss von Beobachtungen außerhalb des aktuellen Suchgebiets reduziert. Dies ermöglicht dem Modell, seine Kapazität gezielter auf die relevantesten Bereiche des Suchraums zu konzentrieren. Wir demonstrieren in synthetischen Optimierungsbenchmarks, dass die gezielte Zugabe von Messunsicherheiten die Optimierungsleistung im Vergleich zu klassischen Transformationen verbessert und einen praxisnahen sowie flexiblen Ansatz zur Erhöhung der Zuverlässigkeit von BO bei Modellabweichungen darstellt.

Contents

1. Introduction

Bayesian optimization (BO) is a popular framework for the global optimization of expensive-to-evaluate black-box functions [12], with applications in robotics [20], control systems [7], and machine learning [36]. It aims to solve the optimization problem by constructing a surrogate model that is leveraged in a sequential decision-making framework. The most widely used surrogate model in BO is the Gaussian process (GP), as it offers compelling flexibility and well-calibrated predictive uncertainties, both of which are crucial for efficient exploration and exploitation [43].

However, the expressiveness of GPs comes with trade-offs that are often overlooked. The strong inductive biases that make GPs effective, such as the kernel choice, also impose potentially restrictive assumptions on the modeled function, namely stationarity, smoothness, and homoscedastic Gaussian noise. These assumptions are rarely fully met in practice [38]. The induced model mismatch can harm the performance of BO. Early observations that inaccurately represent the underlying objective function can adversely influence surrogate hyperparameter learning, leading to suboptimal modeling [22], [45].

Prior work has proposed model-side adaptations such as kernel engineering and warping [35], as well as data-side strategies like output transformations and robust likelihoods to increase robustness [5], [8], [16]. These techniques aim to reduce model mismatch when GP assumptions are violated. More recent work has also introduced point-specific adaptations to mitigate the influence of misleading observations. Relevance-pursuit GPs [3] and divergence-based weighting schemes [2] modify observation weights or noise levels individually to allow the GP to handle heteroscedasticity or sparse corruptions. In parallel, algorithmic adaptations like trust region Bayesian optimization (TuRBO) [8] strengthen optimization efficiency but still rely on standard GPs and remain sensitive to localized model mismatch. Each of the methods listed above has its own limitations.

Despite considerable progress in both model-side and data-side adaptations, there

remains a gap in model fine-tuning at the level of individual observations. Existing literature has primarily focused on global transformations or structural model changes, which do not allow for selectively reducing the influence of outliers that distort the surrogate fit.

Recent methods that focus on point-specific adjustments are still limited to additive noise formulations and usually require additional models. Further, these methods have not been systematically integrated into GP-based BO frameworks.

In this thesis, we aim to address these limitations using targeted noise injection methods to reduce the impact of model mismatch through data-level adjustments. The proposed methods selectively down-weight the influence of misleading observations by injecting observation noise while preserving model fidelity in regions critical to the optimization objective. Specifically, we will introduce a multiplicative Gaussian likelihood that enables observation noise to be scaled individually per training point relative to a globally learned noise level. To identify effective noise configurations, the thesis compares gradient-based optimization of the noise vector with heuristic search strategies, including iterated local search (ILS) and beam search (BS). Moreover, we explore a trust region-based extension that guides noise injection towards observations outside the active trust region and thus improves focus near the optimum.

Our method integrates seamlessly into standard GP-based BO pipelines and is compatible with model-side, data-side, and algorithmic adaptations. In this thesis, we conduct a comprehensive empirical evaluation of the noise injection method using data transformations, standard synthetic benchmarks, multiple acquisition strategies, and BO frameworks, including both vanilla BO and TuRBO.

1.1. Problem Formulation

The overarching goal in this thesis is to identify the global minimizer of an unknown, expensive-to-evaluate black-box function $f : \mathcal{X} \to \mathbb{R}$, defined over a compact domain $\mathcal{X} \subset \mathbb{R}^d$. We can denote this problem as follows:

$$\boldsymbol{x}^\star = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmin}} f(\boldsymbol{x}) \ . \tag{1.1}$$

Since the function f cannot be evaluated directly, we need to rely on noisy observations y in the form:

$$y = f(\boldsymbol{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \ , \tag{1.2}$$

where ε is independent and identically distributed (iid) Gaussian noise with known or learnable variance σ_n^2.

Unlike many BO approaches which assume the objective function f is a sample from a GP, we explicitly refrain from making this assumption. Real-world objective functions frequently violate the smoothness, stationarity, and noise assumptions embedded in common GP priors. We aim to mitigate the negative effects of such model mismatch through data-level adaptation. The effectiveness of our approach, as well as that of the baselines, is evaluated in terms of regret, defined as:

$$r_t = f(\mathbf{x}_t) - f(\mathbf{x}^\star) \ . \tag{1.3}$$

This metric computes the proximity of an observation made at iteration t to the global optimum. An optimizer tries to minimize this regret over a sequence of adaptive evaluations. Depending on the application context, convergence rate and computational efficiency can also be considered important evaluation criteria.

1.2. Key Contributions

The main contributions of this thesis are summarized as follows:

1. **Multiplicative Gaussian Likelihood:** This thesis introduces a novel multiplicative Gaussian likelihood that enables adaptive, per-observation noise scaling in GP regression. Unlike traditional homoskedastic or heteroskedastic noise models, the proposed formulation maintains model tractability while allowing fine-grained control over the influence of individual observations.

2. **Targeted Noise Injection:** Building on the proposed likelihood, this work develops and benchmarks several noise injection strategies tailored for BO. These include gradient-based optimization, discrete heuristic search algorithms, and a spatially guided trust region variant that accumulates noise on persistently irrelevant regions of the input space.

3. **Extensive Empirical Evaluation:** The proposed methods are systematically evaluated on a suite of synthetic optimization benchmarks across varying acquisition functions and input dimensions. In addition to validating the effectiveness of targeted noise injection, the thesis also benchmarks standard output transformations to deepen the mechanistic understanding of how data-side strategies mitigate model mismatch in GP-based BO.

Structure of the Thesis

This thesis is organized into six chapters. Chapter 2 introduces GP, BO, output transformations, and heuristic search algorithms that lay the foundation for the methodological contributions of this thesis. The existing literature pertaining to GP fine-tuning techniques is critically examined in Chapter 3. Methodological innovations, such as the developed targeted noise injection methods combined with the heuristic search and TuRBO algorithms, are detailed in Chapter 4. Chapter 5 presents the empirical results, which include analyses of optimization performance, runtime efficiency, and the patterns of noise injection. Finally, Chapter 6 summarizes the key findings, discusses their implications, and suggests directions for future research.

2. Background

This chapter introduces the mathematical foundations relevant to the methods developed in this thesis to minimize model mismatch. It begins in Section 2.1 with GPs, a class of probabilistic models that define distributions over functions and offer uncertainty estimates. Their flexibility and analytical tractability make them a widely used tool for regression tasks and a foundation for modeling expensive black-box functions. Building on this, the chapter presents in Section 2.2 BO as a sample-efficient strategy for optimizing such functions by iteratively querying the search space based on the predictive distribution of the GP. To reduce the risk of model mismatch, the chapter also introduces output transformations in Section 2.3 as a fine-tuning tool that reshapes observations to better conform to the GP assumptions. Furthermore, selected heuristic search strategies are presented in Section 2.4, which later support the algorithmic adaptation of model behavior. These concepts collectively establish the groundwork upon which the subsequent methodological contributions are built.

2.1. Gaussian Processes

GPs have emerged as a standard modeling choice in probabilistic machine learning due to their closed-form posterior predictive distribution with well-calibrated uncertainty estimates even in the low data regime. A GP defines a distribution over functions, allowing the posterior distribution to be analytically derived after conditioning on observations. These properties make GPs particularly well-suited for sequential decision-making tasks like BO as later discussed in Section 2.2. This section outlines the formal definition of GPs, posterior inference, and hyperparameter estimation, following the textbook by Williams and Rasmussen [43].

Definition and Prior Specification

A GP is a nonparametric prior over functions, which defines a joint distribution over function values at any finite collection of inputs. Formally, a function $f : \mathcal{X} \to \mathbb{R}$ is said to follow a GP if, for any finite set of input points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$, the corresponding function values $\{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)\}$ are jointly multivariate normally distributed. This is denoted as:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')) \ , \tag{2.1}$$

where $m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$ is the mean function and $k(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}'))$ is the covariance or kernel function.

In practice, the mean function $m(\boldsymbol{x})$ is often assumed to be zero without loss of generality, particularly when prior knowledge about the function's behavior is limited. The choice of the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ is more critical, as it encodes assumptions about the smoothness, periodicity, and other structural properties of the target function. The kernel function defines a notion of similarity between inputs, where inputs considered similar by the kernel yield highly correlated function values. This similarity structure governs how information is generalized from observed to unobserved inputs, making the kernel a central modeling component in any GP.

Common kernel choices include the squared exponential or radial basis function (RBF), Matérn, and periodic kernels, each imparting different underlying assumptions. A popular standard choice is the aforementioned RBF kernel, which is denoted as follows:

$$k_{\mathrm{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\ell^2}\right) \ , \tag{2.2}$$

where σ_f^2 denotes the signal variance and ℓ is the characteristic length-scale.

The signal variance governs the overall vertical scale or amplitude of the function values, reflecting how far the function is expected to deviate from the mean. The length-scale, by contrast, controls the degree of wiggliness in the function: smaller values of ℓ allow for more rapid variation in the function, while larger values enforce smoother and more slowly varying functions across the input space.

Despite their flexibility, GPs make strong assumptions that can lead to model mismatch if not fully met, as discussed in Section 1. Some of these assumptions

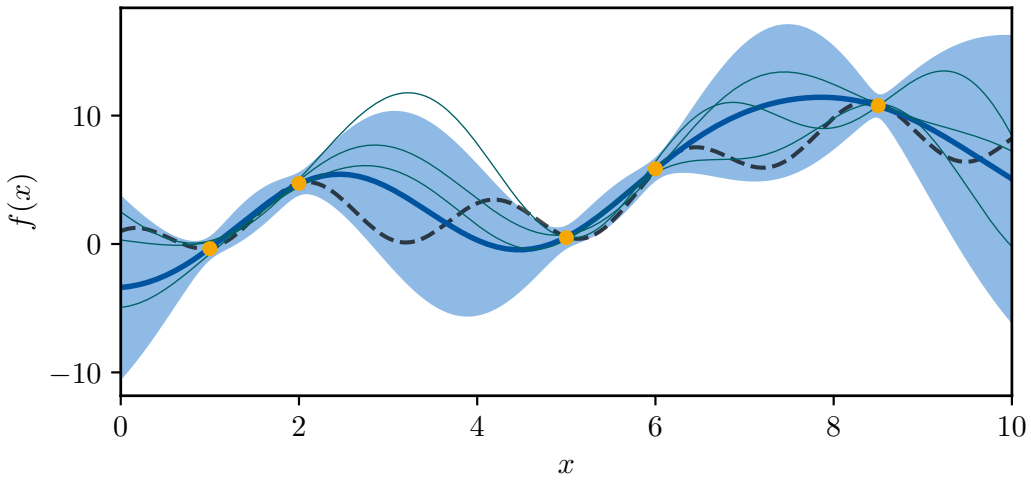

Figure 2.1.: Visualization of a GP posterior after conditioning on observed data (orange dots). The dashed black curve represents the true underlying function $f(x)$. The thick blue line indicates the posterior mean, and the shaded blue region denotes the 95% confidence interval. The thin curves illustrate samples drawn from the GP posterior distribution. The model captures the training data with high confidence and expresses increased uncertainty in regions with fewer observations.

are directly imposed by the kernel choice, which encodes inductive biases such as smoothness and stationarity into the model. These biases determine how the GP generalizes between observations and thereby shape its predictive behavior.

Posterior Inference

Given a GP prior and a set of observed data points $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, one can derive the posterior distribution over the function values. This posterior captures the updated belief about the latent function after incorporating the information from the observations.

As introduced earlier in (1.2), the outputs are assumed to be noisy observations of the latent function, corrupted by independent Gaussian noise with variance σ_n^2. Let $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top$ denote the matrix of the training inputs and $\mathbf{y} = [y_1, \ldots, y_n]^\top$ the corresponding noisy observations.

For a test input $\boldsymbol{x}_*$, the predictive distribution is Gaussian:

$$f(\boldsymbol{x}_*) \mid \mathcal{D}_n \sim \mathcal{N}(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)) \ , \tag{2.3}$$

with closed-form expressions for the posterior mean and variance:

$$\mu(\boldsymbol{x}_*) = \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y} \ , \tag{2.4}$$

$$\sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}_* \ , \tag{2.5}$$

where $\mathbf{k}_* = [k(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}_*, \boldsymbol{x}_n)]^\top$ is the vector of covariances between the test point and training inputs, and $K \in \mathbb{R}^{n \times n}$ is the covariance matrix with entries $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

GPs provide not only point predictions but also principled uncertainty estimates, which are central to their use in BO. As visualized in Figure 2.1, the predictive variance is low near observed data and high in regions far from training points. The ability to perform exact Bayesian inference in closed form is a key strength of GPs, particularly in low-dimensional problems with moderate data sizes, where matrix inversion remains tractable.

Hyperparameter Estimation

GPs rely on a set of hyperparameters that govern both the kernel function and the noise model. These hyperparameters fundamentally shape the behavior of the GP surrogate and directly influence its ability to model functions that are complex, corrupted by noise, or exhibit structured patterns such as periodicity or smooth trends. Common hyperparameters include the signal variance σ_s^2, and the length-scale ℓ, which were previously introduced in the context of the RBF kernel, and the noise variance σ_n^2, which models observation noise. These parameters jointly define the GP prior and the likelihood, and their accurate estimation is essential for predictive performance.

Let $\boldsymbol{\theta}$ denote the vector of all hyperparameters. A principled method for estimating $\boldsymbol{\theta}$ is to maximize the marginal log likelihood (MLL) of the observed data under the GP model. Also known as the model evidence, the marginal likelihood integrates over all latent function values and captures a balance between model fit

and complexity. It is given by:

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log\det(K + \sigma_n^2 I) - \frac{n}{2}\log(2\pi) \ , \quad (2.6)$$

where K is the kernel matrix computed with the current hyperparameter values. The first term measures how well the model explains the observed data and favors hyperparameter settings that lead to high likelihood under the posterior distribution. The second term penalizes overly flexible models by reducing the likelihood of explanations that could plausibly fit a wide range of functions. Finally, the third term serves as a normalization constant.

Hyperparameter estimation is typically performed using maximum likelihood estimation (MLE), which requires gradient-based optimization techniques. In practice, automatic differentiation and modern numerical solvers make this step efficient and scalable, though care must be taken to avoid local optima and ensure numeric stability.

2.2. Bayesian Optimization

BO is a sample-efficient framework for global optimization of expensive and potentially noisy black-box functions. It is particularly suited for settings where each function evaluation involves significant computational or experimental cost, making conventional optimization strategies impractical. BO has been successfully applied across various domains, including control engineering, robotics, chemical design, and clinical trial planning. The method builds on early ideas by Kushner [23] and was later formalized by Mockus [31]. This section largely builds on the recent textbook on BO by Garnett [12].

BO addresses the optimization task defined in Section 1.1, where the goal is to identify the global minimizer of an unknown function. Especially in the context of BO, the function f is assumed to lack a closed-form expression and may only be accessed through point-wise evaluations, each of which may involve substantial computational or experimental costs. Observations are assumed to follow the stochastic model given in (1.2), where each query returns a corrupted version of the true function value due to additive Gaussian noise. To account for this uncertainty, the optimization objective introduced in (1.1) is typically reformulated as minimizing

the expected function value:

$$\boldsymbol{x}^\star = \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}[f(\boldsymbol{x})] \ . \tag{2.7}$$

Here, $\boldsymbol{x}^\star \in \mathcal{X}$ denotes the global minimizer of the expected objective, and $f^\star = \mathbb{E}[f(\boldsymbol{x}^\star)]$ is the corresponding optimal value.

To solve the optimization problem in (2.7), BO employs a sequential optimization strategy consisting of two core components: a probabilistic surrogate model of the unknown objective function and an acquisition function that defines a policy for selecting future evaluation points. This closed-loop strategy enables the algorithm to improve sample efficiency. The framework is particularly suited for optimizing black-box functions, which lack a closed-form expression, provide no gradient information, and can only be accessed through point-wise evaluations. Such functions are common in scientific and engineering domains, where each evaluation may involve computationally expensive simulations, physical experiments, or stochastic procedures.

Surrogate Model

The surrogate model serves as a probabilistic approximation of the unknown objective function, constructed from a finite set of previously observed input–output pairs. Formally, it defines a posterior distribution over functions conditioned on the data, denoted as $f \mid \mathcal{D}_n \sim \mathcal{P}(f \mid \mathcal{D}_n)$.

This probabilistic perspective enables the surrogate to provide not only point estimates of the objective but also uncertainty quantification at unobserved inputs. These uncertainty estimates are essential for guiding the acquisition function, which leverages both predicted objective value and model confidence to select future query points. While various surrogate models are possible, the most widely used in BO are GPs, due to their analytical tractability and expressive nonparametric formulation as introduced in Section 2.1.

Acquisition Functions

The acquisition function $\alpha : \mathcal{X} \to \mathbb{R}$ acts as a decision policy that guides the selection of the next evaluation point in BO. Defined over the same domain as the

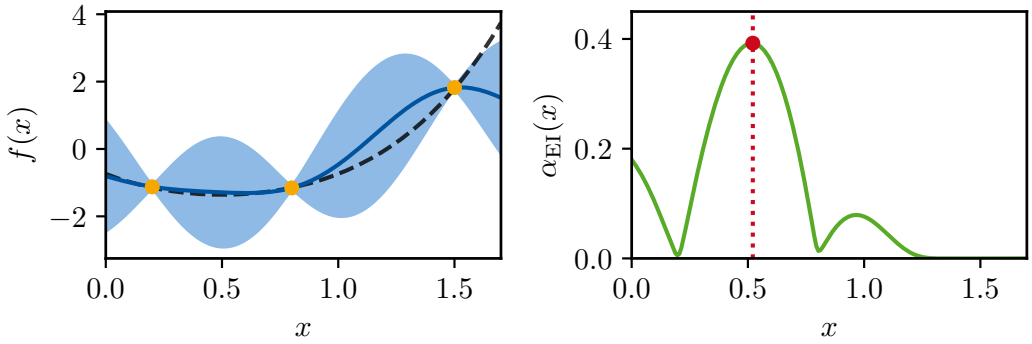

Figure 2.2.: A single iteration of the BO loop. Left: A GP surrogate provides a posterior distribution over the objective function based on prior observations. Right: The expected improvement (EI) acquisition function is formulated from the surrogate's predictions. The next point to evaluate (red dot) is selected at the maximum of the acquisition function, balancing the exploration-exploitation trade-off.

objective function, it is constructed using the surrogate model's predictive mean and uncertainty. At each iteration, the acquisition function assigns a utility value to every candidate input, quantifying the expected benefit of evaluating the objective at that location. While the overall optimization task is formulated as minimization, acquisition functions are still maximized to determine the next query point. To ensure consistency, we negate the acquisition functions when necessary while preserving their standard definitions. Accordingly, the next query point is selected by maximizing the acquisition function:

$$\boldsymbol{x}_{next} = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmax}}\ \alpha(\boldsymbol{x}) \ . \tag{2.8}$$

A central aspect of the acquisition function is the trade-off between exploration and exploitation. Exploration encourages sampling in regions where the surrogate model exhibits high uncertainty, which improves the model's understanding of the function landscape. Exploitation, in contrast, focuses on regions where the model predicts high objective values based on current knowledge. A well-designed acquisition function must strike a balance between these two objectives: overly greedy strategies may converge prematurely to suboptimal regions, while overly exploratory

ones may waste resources in unpromising areas. Different acquisition functions implement this trade-off in distinct ways, some of which are discussed in the following paragraphs.

One widely adopted approach is the EI criterion, which prioritizes points that are expected to yield improvements over the current best observation, denoted f_{best}. Let $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ denote the surrogate model's posterior mean and standard deviation at input $\boldsymbol{x}$, respectively. Under the assumption of Gaussian predictive distributions, the EI is defined as:

$$\alpha_{\text{EI}}(\boldsymbol{x}) = \mathbb{E}\left[\max(0, f(\boldsymbol{x}) - f_{\text{best}})\right] = (\mu(\boldsymbol{x}) - f_{\text{best}})\Phi(z) + \sigma(\boldsymbol{x})\phi(z) \ , \tag{2.9}$$

where $z = \frac{\mu(\boldsymbol{x}) - f_{\text{best}}}{\sigma(\boldsymbol{x})}$, and $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative distribution function (CDF) and probability density function (PDF) of the standard normal distribution, respectively. EI inherently balances the exploration-exploitation trade-off by rewarding points with high predictive mean or high uncertainty. It is widely used in practice due to its smooth optimization surface and reliable behavior across a wide range of problems.

In situations where the objective values are strictly positive and exhibit heavy-tailed or multiplicative noise, applying a logarithmic transformation to the outputs can improve the behavior of the acquisition function. Hutter *et al.* [16] were the first to derive a closed-form expression for EI based on a Gaussian process trained on log-transformed objective values, a variant now commonly referred to as LogEI. By applying the formulation in (2.9) to the logarithm of the response variable, $\ell = \log y$, this approach retains the analytical tractability of classical EI while providing more stable uncertainty estimates in log-space, which in turn improves acquisition decisions in settings with non-Gaussian noise characteristics or large dynamic ranges.

Alternatively, the upper confidence bound (UCB) acquisition function selects points based on an optimistic estimate of the objective, defined as:

$$\alpha_{\text{UCB}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \beta \cdot \sigma(\boldsymbol{x}) \ , \tag{2.10}$$

where $\beta > 0$ is a tunable hyperparameter that determines the trade-off between exploration and exploitation. Larger values of β promote exploration by favoring regions with high predictive uncertainty, while smaller values bias the selection to-

ward areas with high predicted means. Due to its analytical simplicity and flexibility, UCB remains a widely used and interpretable strategy in BO.

Algorithm

With the core components of a surrogate model and an acquisition function, the full sequential optimization procedure of BO can now be formalized. Starting with an initial dataset $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$, the algorithm proceeds by alternating between model fitting and acquisition optimization.

At iteration t, the surrogate model is refit to the current dataset $\mathcal{D}_t$, and the next query point is selected by maximizing the acquisition function as described in (2.8).

The black-box objective function is then evaluated at $\boldsymbol{x}_t$, yielding a possibly noisy observation $y_{t+1} = f(\boldsymbol{x}_t) + \varepsilon$, which is appended to the dataset: $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\boldsymbol{x}_t, y_t)\}$. This process continues until a predefined termination condition is satisfied.

The vanilla BO procedure is summarized in Algorithm 1.

Algorithm 1 Vanilla Bayesian Optimization

procedure BayesianOptimization(f, $\mathcal{X}$, $\mathcal{D}_0$, budget)
    Initialize dataset: $\mathcal{D}_0 \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0}$
    for $t = 0$ to budget -1 **do**
        Fit a probabilistic surrogate model to $\mathcal{D}_t$ (e.g., Gaussian Process)
        Select next input: $\boldsymbol{x}_t \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{X}} \alpha(\boldsymbol{x}; \mathcal{D}_t)$
        Evaluate objective: $y_t \leftarrow f(\boldsymbol{x}_t) + \varepsilon$
        Update dataset: $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(\boldsymbol{x}_t, y_t)\}$
    end for
    return $\mathcal{D}_{\text{final}}$
end procedure

This basic loop can be extended to accommodate batched evaluations, multi-fidelity models, or adaptive acquisition strategies, depending on the specific application.

2.3. Output Transformations

Output transformations are widely used to improve the robustness and calibration of GP models to reduce model mismatch. These transformations reshape the dis-

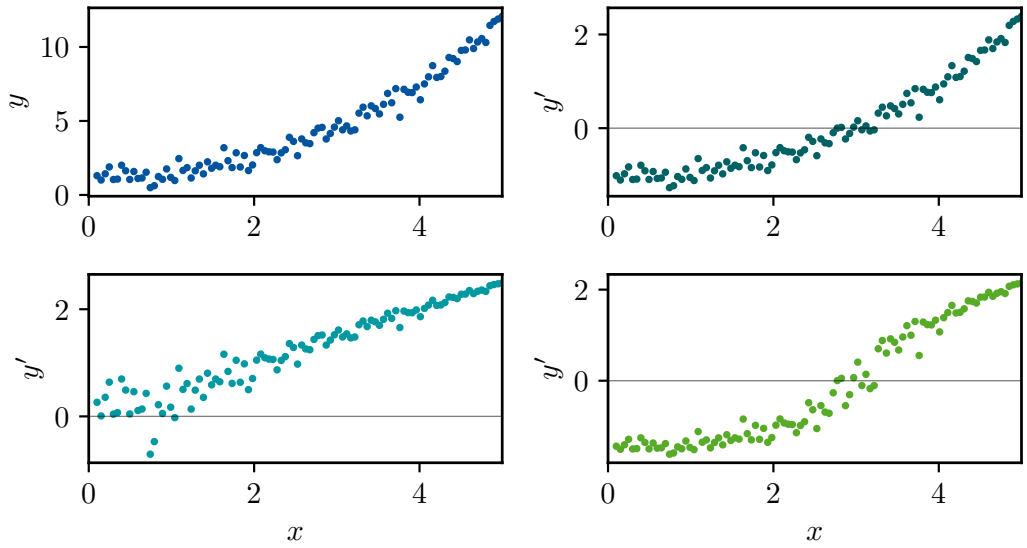

Figure 2.3.: The effect of common output transformations on a synthetic dataset exhibiting a nonlinear trend and constant variance. Top left: Original data with positive outputs. Top Right: Standardization rescales the data to have zero mean and unit variance. Logarithmic transformation compresses high values, reducing skew for positive-only data. Bottom Right: A bilogarithmic transformation symmetrically compresses both positive and negative tails of the distribution after mean-centering of the original data to introduce negative values.

tribution of the target variable to better satisfy the fundamental GP assumptions. While broader families of transformations exist, we focus here on the three variants most frequently employed in BO. This section outlines the mathematical foundation of these transformations and briefly discusses their properties.

Standardization

Standardization is the most commonly used transformation. It rescales the output to have a mean of zero and a unit variance. Formally, the transformation is denoted as:

$$y' = \frac{y - \mu_y}{\sigma_y} \,, \tag{2.11}$$

where μ_y and σ_y denote the empirical mean and standard deviation of the observed outputs. Standardization improves numerical stability during GP fitting, reduces sensitivity to the scale of the output, and is particularly beneficial when the GP kernel is not invariant to output scaling. It also accelerates convergence during hyperparameter optimization and is often applied as a preprocessing step [43].

Logarithmic Transformation

In many real-world scenarios, especially those with strictly positive and right-skewed outputs, a logarithmic transformation can render the data more homoscedastic and approximately Gaussian. The log transform is, in fact, the limiting case $\lambda \to 0$ of the broader family of power transformations first formalised by Box and Cox [5]:

$$y' = \log(y + c), \qquad c > 0 \ , \tag{2.12}$$

where the constant c guarantees numerical stability when y approaches zero. By compressing large values and expanding small ones, the transformation stabilizes the variance of heavy-tailed responses [16].

Bilogarithmic Transformation

The bilogarithmic transformation is an extension of the logarithmic transformation to data with mixed signs, defined as:

$$y' = \text{sign}(y) \cdot \log(|y| + 1) \ . \tag{2.13}$$

It corresponds to the signed absolute logarithmic transformation from (2.12) with $c = 1$. The transformation is symmetric around zero and maintains the benefits of variance stabilization from the logarithmic transformation by stretching values near zero and compressing extreme ones. Additionally, it takes positive and negative values [8].

2.4. Heuristic Search Algorithms

Heuristic search algorithms are commonly used in optimization and artificial intelligence to efficiently navigate large or complex search spaces without the need to

evaluate every possible outcome. In this work, we will leverage such strategies to identify scaling noise vectors for our newly proposed likelihood, as introduced in Section 4.1.

Iterated Local Search

ILS functions as a metaheuristic framework designed to enhance traditional local search methods through the introduction of systematic perturbations. Through the modification of solution components, these perturbations enable the process to break free from local optima while still preserving parts of the current promising solution to guide the search. The method functions through a repetitive sequence of three fundamental steps where initially a perturbation modifies a subset of the current solution followed by its refinement via a local search procedure. The local search procedure is problem-specific and can be tailored accordingly. It ends with an acceptance decision mostly based on an objective value to retain only those solutions which show progress for continued investigation. This iterative process continues until a stopping condition is reached, such as a set time limit or a maximum number of iterations [28]. The overall procedure is summarized in Algorithm 2.

Algorithm 2 Iterated Local Search

Require: Initial solution s_0, perturbation operator Perturb, local search procedure LocalSearch, objective function f, termination criterion

1: $s^\star \leftarrow s_0$
2: **while** termination criterion not met **do**
3: // **1. Perturbation**
4: $s' \leftarrow \text{Perturb}(s^\star)$
5: // **2. Local Improvement**
6: $s'' \leftarrow \text{LocalSearch}(s')$
7: // **3. Acceptance Criterion**
8: **if** $f(s'') > f(s^\star)$ **then**
9: $s^\star \leftarrow s''$
10: **end if**
11: **end while**
12: **return** $s^\star$

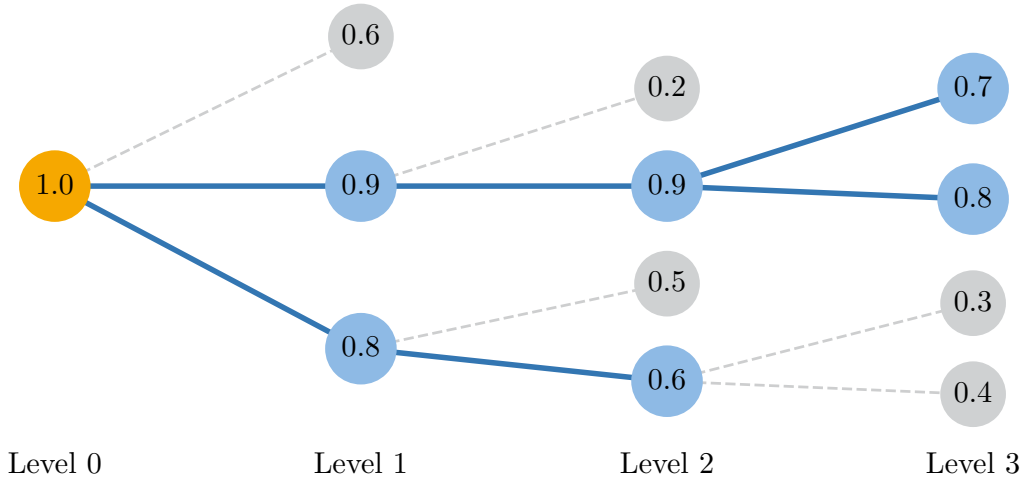

Figure 2.4.: Visualization of BS with beam width $k = 2$. The figure illustrates a tree-based search process across four levels. Nodes represent candidate solutions, annotated with their evaluation scores. At each level, only the top-k scoring nodes (in blue) are retained and expanded, while lower-scoring candidates (in gray) are pruned from the search. The search begins from the root node (orange) at Level 0 and proceeds by selecting the top candidates at each step.

Beam Search

BS is a heuristic tree search algorithm designed to balance exploration breadth with computational efficiency by restricting the number of candidate expansions at each depth level. At each iteration, the algorithm maintains a fixed number of the most promising partial solutions, ranked according to a heuristic, which are denoted by the beam width k. Then only these top-k candidates are expanded in the next iteration to generate the new level of solutions [29]. This makes BS a computationally tractable alternative to exhaustive breadth-first search, particularly in high-dimensional or combinatorial settings [10]. Compared to purely greedy strategies, which expand only the single best candidate at each step, BS retains diversity in the search process by simultaneously considering multiple high-quality paths. Extensions such as Diverse Beam Search explicitly promote diversity among candidate beams to avoid generating similar beams [41]. The basic BS principle is visualized in Figure 2.4.

3. Related Work

Building on the motivation established in Chapter 1, this chapter examines established strategies for addressing model mismatch arising from violated GP assumptions. Prior literature reduces this model mismatch through adaptations to the surrogate model, the training data, or the algorithmic structure of BO, as detailed in Sections 3.1, 3.2, and 3.3, respectively. The following review positions the contribution of the thesis, namely targeted noise injection, within that landscape.

3.1. Model-Side Adaptations

Model-side adaptations seek to modify the internal structure of the GP so that its underlying assumptions are more closely satisfied.

Kernel Engineering. Deep kernel learning replaces the stationary kernel with a neural feature extractor, which models rich non-stationary patterns. Despite increasing expressiveness, it offers no mechanism for down-weighting specific outliers [44].

Warped GPs. The work of Snelson *et al.* [35] proposes the learning of a monotone output warp jointly with kernel hyperparameters. Snoek *et al.* [37] extend warping to the input space for BO on non-stationary functions. Both apply a single global transformation; consequently, corrupted points can still distort the model.

Meta-learned and Pre-trained GPs. HyperBO transfers kernel priors across tasks to speed adaptation [42]. Such methods assume a pool of related tasks and do not address task-specific heteroscedastic noise.

Change Point Detection. Posterior tempering detects abrupt regime shifts and down-weights obsolete data [26]. It excels at temporal drift when an underlying distribution has changed but is less effective for sparse, local anomalies.

Overall, model-side adaptations often require complex architectures or strong prior assumptions and remain vulnerable to local outliers. Our approach addresses this limitation by down-weighting specific observations without modifying the surrogate

model. Therefore, targeted noise injection could be combined with existing model-side adaptations.

3.2. Data-Side Adaptations

In contrast to modifying the GP architecture itself, data-side adaptations target the training data and the observation model directly.

Data Transformation. As outlined in Section 2.3, output transformations are commonly used to improve alignment with GP assumptions by stabilizing variance or reducing skewness [8], [16]. Standardization, in particular, is frequently adopted as a default baseline in prior work [4]. Such global transformations can attenuate the effect of outliers to some extent, especially when applied to heavy-tailed responses. Nevertheless, they act uniformly on all observations and therefore lack the flexibility to selectively adjust the influence of a single misleading data point.

Robust Likelihoods and Sample Re-weighting. Student-t likelihoods offer a globally robust treatment of heavy-tailed noise by attenuating outlier influence via a heavy-tailed observation model [21]. Relevance-pursuit GPs introduce pointwise noise variances by sequentially maximizing the marginal likelihood, yet adhere to an additive noise model and require additional per-point parameters [3]. Altamirano *et al.* [2] use β-divergence to assign automatic, closed-form weights that shrink the influence of low-likelihood observations in GP regression. Heteroscedastic GP models extend this paradigm by learning input-dependent noise via a secondary GP or variational inference but maintain an additive noise assumption [24], [25].

Data Augmentation. Recent work synthesizes pseudo-observations from the GP posterior to improve coverage [39]. Augmentation adds information but does not temper already collected outliers, hence complementing rather than reweighting.

Data-side methods like output transformations and robust likelihoods typically apply global changes. The existing point-specific techniques use additive, observation-specific noise terms. Our contribution can similarly be classified as a data-side adaptation, as it introduces a novel likelihood function designed to adjust the global signal noise scale on an individual point basis in a multiplicative manner. Due to its model-agnostic nature, our method can be readily combined with standard data transformations to improve overall robustness.

3.3. Algorithmic Adaptations

An additional line of research explores adaptations that modify the BO algorithm itself. Representative examples include batch acquisition strategies [13], asynchronous BO [19], multi-fidelity modeling [9], and trust-region-based techniques [8]. In this thesis, we propose to extend our newly proposed method with the ladder adaptation.

Trust Region Bayesian Optimization. TuRBO is an algorithmic framework designed to improve the scalability and robustness of BO in high-dimensional settings. It achieves this by partitioning the search space into dynamically adjusted trust regions, within which local surrogate models are optimized independently. The acquisition function then samples from these regions [8].

Nevertheless, TuRBO still fundamentally relies on the underlying GP assumptions within its trust regions. This reliance makes it susceptible to model mismatch inside the trust regions. By explicitly integrating targeted noise injection with TuRBO's algorithmic framework, the methods proposed in this thesis seek to directly control model mismatch within those trust regions.

4. Targeted Noise Injection

Motivated by the limitations of current data-side adaptations discussed in Chapter 3, this chapter introduces a novel methodology to selectively reduce the influence of misleading training points in GP-based BO. The central contribution is a targeted noise injection framework that adjusts the observation noise at the level of individual training points. This is enabled by a newly proposed multiplicative Gaussian likelihood, which allows per-point scaling of the global noise level without altering the underlying GP model structure.

Section 4.1 introduces this likelihood formulation and outlines its integration into standard GP regression. The subsequent sections detail three complementary strategies for identifying effective noise scaling vectors required by the presented likelihood formulation. First, we present a gradient-based optimization method in Section 4.2. Second, in Section 4.3, we discuss a set of heuristic search algorithms, and finally, a trust-region-guided scheme to inject noise outside a trust region is presented in Section 4.4.

Throughout this chapter, we adopt the standard assumption that maximizing the MLL of the GP surrogate yields improved model calibration and thus better BO performance. This assumption aligns with the empirical Bayes framework introduced in Section 2.1, where hyperparameters are optimized via marginal likelihood. Prior work supports this approach: Hvarfner *et al.* [17] and Xu *et al.* [46] show that carefully fitted vanilla GPs can match or outperform more complex surrogates. Moreover, recent robustness extensions, such as the relevance-pursuit GP of Ament *et al.* [3], similarly rely on marginal likelihood maximization to assign point-specific noise levels and thus underscore the general applicability and centrality of this objective.

4.1. Multiplicative Gaussian Likelihood

To enable selective down-weighting of individual training points in Gaussian process regression, we introduce a new multiplicative Gaussian likelihood that assigns a learnable multiplicative noise factor $\lambda_i > 0$ to each observation. These factors scale the global noise variance σ_n^2 on a per-point basis, leading to the following noise model:

$$\varepsilon_i \sim \mathcal{N}(0, \lambda_i \cdot \sigma_n^2)\,. \tag{4.1}$$

This formulation retains the standard Gaussian noise model as described in (1.2) when $\lambda_i = 1$ for all i, but enables targeted attenuation of specific data points by reducing individual λ_i values.

Assuming independence across observations, the overall likelihood becomes

$$p(\mathbf{y} \mid \mathbf{f}) = \prod_{i=1}^{n} \mathcal{N}(y_i \mid f_i, \lambda_i \cdot \sigma_n^2)\,, \tag{4.2}$$

and the corresponding MLL is given by:

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\lambda}) = -\frac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 \Lambda)^{-1}\mathbf{y} - \frac{1}{2}\log\det(K + \sigma_n^2\Lambda) - \frac{n}{2}\log(2\pi)\,, \tag{4.3}$$

where $\Lambda = \operatorname{diag}(\boldsymbol{\lambda})$ is the diagonal matrix of the multiplicative noise vector. This preserves the analytical tractability of the GP framework and allows the likelihood to be optimized with standard MLL maximization.

The noise vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ is constrained to lie within the closed interval $[\epsilon_{\min}, 1]$, where $\epsilon_{\min}$ is a small positive constant introduced for numerical stability. Throughout this work, we fix $\epsilon_{\min} = 10^{-6}$. The upper bound $\lambda_i = 1$ corresponds, when applied to all points uniformly, to the standard homoskedastic noise model [43]. Values of λ_i below 1 decrease the observation noise for individual data points. Alternatively, increasing λ_i injects additional noise and therefore down-weights the influence of the corresponding observation in the MLL optimization of the GP and thus, the GP hyperparameter learning. Allowing $\lambda_i > 1$ could lead to ill-conditioned covariance matrices $K + \sigma_n^2\Lambda$, as disproportionately large diagonal entries introduce scale imbalances between observations. This degrades both numerical stability and the accuracy of matrix inversion during inference. Importantly, the global noise variance σ_n^2 remains a learnable hyperparameter. Hence, uniform increases in λ_i

across all data points would be offset by a corresponding decrease in σ_n^2, limiting the practical effect of such scaling and reinforcing the utility of localized adjustments.

The multiplicative Gaussian likelihood offers a more flexible alternative to homoskedastic and heteroskedastic noise assumptions. Homoskedastic likelihoods share a global noise variance, whereas heteroskedastic formulations assign individual noise variances $\sigma_{n,i}^2$ to each observation [14]. This is typically achieved at the cost of increased model complexity or the introduction of auxiliary models [24]. When noise levels are not inferred from data, they must be defined manually based on prior knowledge, which may not be available or reliable. In contrast, the multiplicative formulation maintains an overall scale but permits per-point reweighting within the range defined by the learned scaling factor. Existing frameworks such as `BoTorch` [4] offer support for heteroskedastic noise learning via additive terms, but do not natively support multiplicative scaling. Table 4.1 summarizes the key differences.

Table 4.1.: Comparison of Gaussian likelihood formulations.

Formulation	**Noise Expression**	**Adaptivity Mechanism**
Homoskedastic	$\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$	shared scalar
Heteroskedastic	$\varepsilon_i \sim \mathcal{N}(0, \sigma_{n,i}^2)$	individual variance terms
Multiplicative	$\varepsilon_i \sim \mathcal{N}(0, \lambda_i \cdot \sigma_n^2)$	scaling of shared noise

The noise vector λ can either be fixed in advance or adapted during training. In the adaptive case, they may be optimized using gradient-based methods or heuristic search strategies. This flexibility enables both the static injection of prior knowledge and the dynamic reweighting of observations based on feedback from the optimization process. The following sections investigate both approaches.

4.2. Gradient-Based Noise Injection

To establish a reference point for the heuristic and trust-region methods evaluated later in this chapter, we consider joint optimization of the per-observation noise scaling factors $\boldsymbol{\lambda}$ alongside the standard GP hyperparameters. Since MLL maximization is the standard criterion for hyperparameter learning in GPs [43], extending it to include $\boldsymbol{\lambda}$ provides a natural and principled baseline. Although this approach may be computationally demanding, it provides a valuable benchmark for assessing

whether more efficient noise injection strategies can approximate the behavior of fully optimized solutions. Because the MLL is generally non-convex in both the kernel hyperparameters and the noise factors, joint optimization may converge to suboptimal local optima, particularly in high-dimensional or noisy settings.

Continuous Factor Optimization

In the continuous setting, the noise scaling factors $\lambda_i \in [\epsilon_{\min}, 1]$ are modeled as differentiable parameters and optimized together with the GP hyperparameters:

$$\boldsymbol{\lambda}^*, \boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\lambda}, \boldsymbol{\theta}} \log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\lambda}) \ . \tag{4.4}$$

This continuous variant enables fine-grained control of the noise allocation across training points and serves as the most expressive variant among the optimization-based strategies. The approach is referred to as gradient-based noise injection (GB-NI). Further algorithmic details can be found in the Appendix A.1.

Binary Factor Optimization

In addition to the continuous formulation, we investigate a binary variant of the noise injection scheme to align more closely with the structure of later heuristic strategies. This motivates a binary optimization baseline in which each observation is either assigned full noise ($\lambda_i = 1$) or minimal noise ($\lambda_i = \epsilon_{\min}$), denoted binary gradient-based noise injection (B-GB-NI).

To enable gradient-based optimization despite the binary nature of the noise assignments, we adopt a noise-free Binary-Concrete relaxation of a Bernoulli gate [18], [30]. Concretely, we parametrize each factor with a logit η_i and set:

$$\lambda_i = \sigma(\eta_i) \cdot 1 + (1 - \sigma(\eta_i)) \cdot \epsilon_{\min} \ , \tag{4.5}$$

where $\sigma(\eta_i) = (1 + e^{-\eta_i})^{-1}$. The affine stretch to the interval $[\epsilon_{\min}, 1]$ mirrors the hard-Concrete gate of Louizos *et al.* [27], and the relaxed factors remain fully differentiable, allowing the logits to be optimized jointly with GP hyperparameters via the MLL. At convergence, binary decisions about noise injection per point are recovered by thresholding. Additional details can again be found in the Appendix A.1.

4.3. Heuristic Noise Injection

Although gradient-based methods enable principled MLL maximization, they are prone to local optima and sensitive to hyperparameter initialization, particularly in non-convex or noisy settings. Their continuous parameterization also restricts them to smooth, local adjustments, which may fail to uncover structurally meaningful configurations of noise allocation.

To confront these limitations, we introduce a suite of heuristic search algorithms that treat noise injection as a combinatorial optimization problem over binary noise assignments. Each observation is assigned either standard noise ($\lambda_i = 1$) or minimal noise ($\lambda_i = \epsilon_{\min}$), allowing the algorithms to operate in a discrete search space. This framing enables the use of well-established metaheuristic strategies such as greedy search, ILS, and BS. See Appendix B for an overview of global hyperparameters for the subsequent algorithms.

Beyond their computational advantages, heuristic methods can uncover interpretable structure in the learned noise patterns that may remain hidden in continuous optimization. By traversing diverse configurations, they can surface consistent down-weighting of certain points, revealing potential sources of model mismatch or outlier influence. In this sense, heuristic noise injection serves not only as a practical alternative but also as a diagnostic lens into the structure and reliability of the GP model under noisy or misspecified conditions.

4.3.1. Hyperparameter Optimization Strategies

A critical aspect of heuristic noise injection is the choice of global configuration strategy, which includes both the treatment of GP hyperparameters and the initialization of the noise vector.

Pre-fit Strategy. In the pre-fit strategy, GP hyperparameters are optimized once using a standard homoskedastic Gaussian likelihood before the noise factor search begins. These kernel hyperparameters (length-scales, signal variance) and the global noise variance are then fixed throughout the search. This approach offers substantial computational efficiency, as MLL evaluations avoid the overhead of repeated hyperparameter tuning. It also improves algorithmic stability by ensuring that likelihood comparisons reflect genuine differences in noise configurations rather than artifacts from re-optimization. The computational cost scales linearly with the number of fac-

tor configurations considered, making this strategy well-suited for high-dimensional problems where repeated GP training becomes prohibitive.

Re-fit Strategy. The iterative refit strategy re-optimizes all hyperparameters for each candidate noise configuration before evaluating the MLL. While computationally expensive, this ensures that every factor assignment is evaluated under its optimal model fit, potentially yielding more accurate likelihood estimates and stronger final performance.

Noise Vector Initialization. In addition to the choice of how hyperparameters are treated, each heuristic algorithm can be initialized using one of two noise configurations. In the noise-first variant, the noise vector is initialized with $\boldsymbol{\lambda} = \mathbf{1}_n$, assigning high noise to all points. In contrast, the no-noise-first variant starts with $\boldsymbol{\lambda} = \epsilon_{\min} \cdot \mathbf{1}_n$, assuming all points are initially trustworthy. These initialization schemes interact with the pre-fit and refit strategies to form four distinct global configurations applicable across all heuristic methods.

4.3.2. Naive Noise Injection

The naive noise injection (Naive-NI) algorithm represents the simplest heuristic for optimizing per-observation noise factors. It evaluates each data point independently to decide whether applying minimal noise ($\lambda_i = \epsilon_{\min}$) improves the MLL relative to retaining standard noise ($\lambda_i = 1$). While this greedy, pointwise procedure ignores interdependencies between observations, its simplicity results in low computational overhead and fast runtime.

Algorithm 3 Naive Noise Injection Algorithm (default: pre-fit & noise-first variant)

Initialize $\boldsymbol{\lambda} \leftarrow \mathbf{1}_n$ ▷ start with all noise
for $i = 1$ to n **do** ▷ n: number of training points
  $\lambda_i \leftarrow 1.0$; $L_{\text{noise}} \leftarrow \text{MLL}(\boldsymbol{\lambda})$
  $\lambda_i \leftarrow \epsilon_{\min}$; $L_{\min} \leftarrow \text{MLL}(\boldsymbol{\lambda})$
  $\lambda_i \leftarrow \begin{cases} 1.0 & \text{if } L_{\text{noise}} > L_{\min} \\ \epsilon_{\min} & \text{otherwise} \end{cases}$
end for
return $\boldsymbol{\lambda}$

Algorithm 3 outlines the noise-first variant using the pre-fit strategy, which serves as the canonical baseline. For each observation, the algorithm compares the marginal

log-likelihood under both noise levels and selects the setting with higher likelihood. This results in a locally optimal noise configuration under the assumption of conditionally independent updates.

4.3.3. Iterated Local Search Noise Injection

To extend the search horizon beyond the local optima typically reached by greedy methods, we adapt the general ILS metaheuristic framework [28], introduced in Section 2.4, to the noise injection setting. We refer to this adaptation as iterated local search noise injection (ILS-NI).

In our adaptation of Algorithm 2, the perturbation step randomly selects a fixed proportion of observations. Each candidate in the perturbed subset undergoes local search using the naive evaluation procedure described in Section 4.3.2. This yields a new optimized configuration for the subset and results in a new noise vector. If this noise vector improves the MLL upon the best solution so far, it is accepted and replaced as the new best solution. The corresponding implementation is detailed in Algorithm 4 and represents the pre-fit variant initialized with noise-first.

Algorithm 4 ILS Noise Injection Algorithm (pre-fit, noise-first variant)

Require: Perturbation size p, max iterations T

Initialize $\boldsymbol{\lambda} \leftarrow \mathbf{1}_n$ ▷ start with all noise
$L^\star \leftarrow \mathrm{MLL}(\boldsymbol{\lambda}^\star)$
for $j = 1$ to T **do**
  $m \leftarrow \max(1, \lfloor p \cdot n \rfloor)$
  $\mathcal{I} \leftarrow \mathrm{randomSubset}(\{1, \ldots, n\}, m)$; $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^\star$ ▷ **/* Perturbation */**
  for $i \in \mathcal{I}$ **do** ▷ **/* Local improvement */**
    $L_1 \leftarrow \mathrm{MLL}(\boldsymbol{\lambda} \text{ with } \lambda_i = 1)$
    $L_\epsilon \leftarrow \mathrm{MLL}(\boldsymbol{\lambda} \text{ with } \lambda_i = \epsilon_{\min})$
    $\lambda_i \leftarrow \begin{cases} 1 & \text{if } L_1 > L_\epsilon \\ \epsilon_{\min} & \text{otherwise} \end{cases}$
  end for
  $L \leftarrow \mathrm{MLL}(\boldsymbol{\lambda})$
  if $L > L^\star + 10^{-6}$ **then** ▷ **/* Acceptance */**
    $\boldsymbol{\lambda}^\star \leftarrow \boldsymbol{\lambda}$; $L^\star \leftarrow L$
  end if
end for
return $\boldsymbol{\lambda}^\star$

Importantly, unlike the naive algorithm in Section 4.3.2, which evaluates each point in isolation and accepts changes immediately, the ILS method applies updates in batch to a group of noise factors, evaluates the resulting joint configuration and accepts it only if it yields a global improvement. This distinction enables the method to escape poor local configurations by evaluating higher-order dependencies between observations. To improve runtime efficiency, an early stopping mechanism is also incorporated which detects if a plateau is reached.

4.3.4. Beam Search Noise Injection

To further improve exploration of the discrete noise configuration space, we adapt the general BS framework [29], introduced in Section 2.4, to the noise injection setting. We denote this variant as beam search noise injection (BS-NI). BS incrementally builds up candidate solutions by expanding a fixed number of the most promising noise configurations at each depth level, thereby balancing exploration breadth with computational efficiency.

Algorithm 5 Beam Search Noise Injection Algorithm (pre-fit, noise-first variant)

Require: Beam width k, maximum iterations T
$\boldsymbol{\lambda}^{(0)} \leftarrow \mathbf{1}_n$ ▷ start with all noise
$L^{(0)} \leftarrow \mathrm{MLL}(\boldsymbol{\lambda}^{(0)})$
$\mathcal{B} \leftarrow \{(\boldsymbol{\lambda}^{(0)}, L^{(0)})\}$ ▷ current beam
for $j = 1$ to T **do**
    $\mathcal{C} \leftarrow \emptyset$ ▷ candidate pool
    /* Beam Expansion */
    for each $(\boldsymbol{\lambda}, L)$ in $\mathcal{B}$ **do**
        for each index i with $\lambda_i = 1$ **do**
            $\boldsymbol{\lambda}' \leftarrow \boldsymbol{\lambda}; \quad \lambda'_i \leftarrow \epsilon_{\min}$
            $L' \leftarrow \mathrm{MLL}(\boldsymbol{\lambda}')$
            if $L' > L$ **then**
                add $(\boldsymbol{\lambda}', L')$ to $\mathcal{C}$
            end if
        end for
    end for
    Select top-k candidates from $\mathcal{C}$ by MLL score to form next beam $\mathcal{B}^{(j+1)}$
end for
return $\arg \max_{(\boldsymbol{\lambda}, L) \in \mathcal{B}} L$

In our adaptation of Algorithm 5, each beam element represents a distinct binary noise configuration vector $\boldsymbol{\lambda} \in \{\epsilon_{\min}, 1\}^n$. Starting from a noise-first or no-noise-first configuration, each iteration expands the current beam by flipping a single noise factor (i.e., from 1 to $\epsilon_{\min}$), evaluating the resulting configuration via the MLL, and retaining only the top-k candidates. Redundant configurations are pruned to ensure that only unique candidate noise vectors are evaluated at each iteration. The search terminates either when no new candidates are found or a maximum depth is reached.

The corresponding implementation is shown in Algorithm 5 and reflects the pre-fit variant initialized with the default all-noise configuration. Unlike the naive method (Section 4.3.2), which evaluates each point in isolation and greedily accepts local improvements, BS explores sequences of factor changes over multiple steps, conditioning future decisions on previously selected configurations. This allows it to capture higher-order dependencies between noise assignments that would be missed by pointwise methods. Compared to ILS, which perturbs and refines subsets in parallel through stochastic restarts, BS performs deterministic and structured exploration along the most promising trajectories. As a result, it is particularly well-suited for uncovering cumulative or interacting effects in complex noise landscapes.

4.4. Trust Region Noise Injection

BO methods typically rely on fitting a global surrogate model to all observed data, even though only a small subset of these observations may be relevant to the current optimization phase. This mismatch between model scope and local search intent can degrade performance, especially in the presence of noisy or misleading observations [17]. To address this limitation, we introduce trust region noise injection, a methodologically novel approach that leverages spatial locality to selectively reduce the influence of observations located outside the current region of interest.

Our approach draws on the TuRBO framework [8], which restricts the acquisition function to a local neighborhood around the best solution found so far. We extend this idea to the data side by assigning lower observation noise within the trust region ($\lambda_i = \epsilon_{\min}$) and higher noise outside ($\lambda_i = 1$). The surrogate model is thus encouraged to prioritize fidelity where optimization is actively occurring while remaining flexible elsewhere. Importantly, by forcing the model to fit observations more precisely inside the trust region, our approach enables more accurate learning

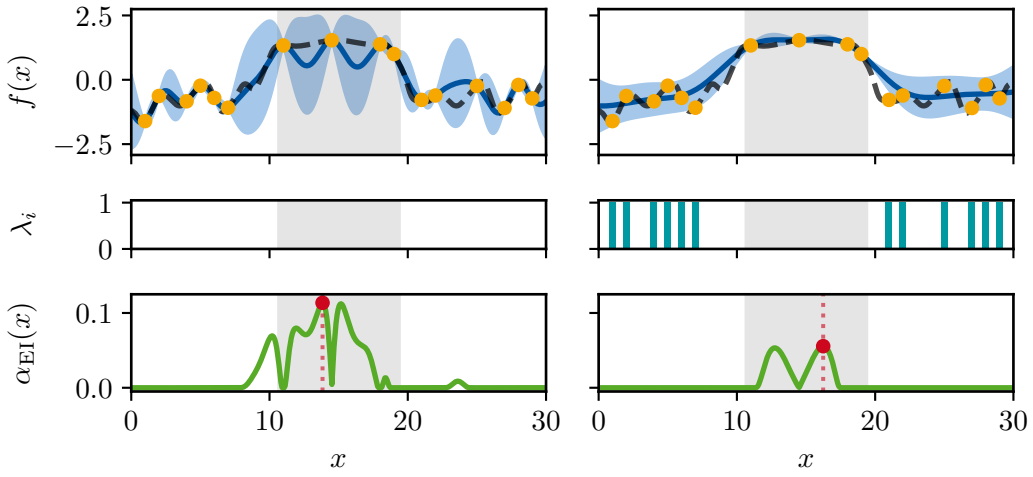

Figure 4.1.: This figure illustrates the effect of trust region noise injection. Left: A standard GP, which weights all data uniformly, produces a compromised posterior fit within the active trust region (gray area) due to influential outlying observations. Right: The proposed method injects noise into observations outside the trust region, down-weighting their influence. This results in a more accurate local posterior and a more focused acquisition function. Panels depict the GP posterior (top), per-point noise allocation λ_i (middle), and the resulting EI acquisition function (bottom).

of local hyperparameters that govern the behavior of the acquisition function. This effect is illustrated in Figure 4.1, which contrasts uniform and localized noise allocation strategies. In particular, the left panel demonstrates that uniformly weighting all observations may distort length-scale estimation, leading to overly short and oscillatory fits, whereas the right panel shows how spatially selective noise allocation yields smoother, better-calibrated posterior estimates by focusing model capacity on locally relevant data.

This chapter presents two variants of trust region noise injection. First, we provide theoretical and algorithmic background on TuRBO to motivate the role of spatial constraints. Then, we introduce a trust-region-guided noise injection model, which receives trust region boundaries from an external optimizer and adjusts noise allocations accordingly. We distinguish between a binary variant that injects noise only when it improves the MLL and an additive variant that accumulates noise across

iterations to reflect persistent irrelevance. We then introduce a BS variant that unifies these ideas into a more expressive decision-making framework.

4.4.1. Trust Region Bayesian Optimization

TuRBO [8] is a scalable BO framework designed to address the limitations of global surrogate modeling in high-dimensional settings. Rather than optimizing the acquisition function over the entire input space, TuRBO confines the search to localized trust regions—subdomains in which the surrogate model is expected to make reliable predictions. Restricting the search to trust regions improves sample efficiency and reduces the risk of poor extrapolation.

The core mechanism in TuRBO is the dynamic adjustment of a hyperbox-shaped trust region centered at the current best solution. At iteration t, the region is defined as

$$\mathcal{TR}^{(t)} = \left\{ \boldsymbol{x} \in \mathcal{X} \;\middle|\; \left\| \boldsymbol{x} - \boldsymbol{x}_c^{(t)} \right\|_\infty \leq \frac{\tau^{(t)}}{2} \right\} , \tag{4.6}$$

where $\boldsymbol{x}_c^{(t)} \in \mathbb{R}^d$ is the center, typically the best observed point, and $\tau^{(t)} > 0$ denotes the side length of the axis-aligned region. The region expands following successful evaluations and contracts otherwise. This adaptation is governed by a pair of counters for success and failure, compared against thresholds T_{succ} and T_{fail}, respectively. In practice, the trust region is often rescaled using the surrogate's learned length-scales to reflect anisotropic sensitivity. The full rescaling procedure is detailed in Appendix A.2.

Two main variants of TuRBO exist. TuRBO-1 maintains a single trust region and is suited for scenarios where the global optimum is believed to lie within a compact subset of the domain. TuRBO-m maintains m parallel regions with independent GPs, enabling greater exploration and parallel evaluations. In this work, we adopt TuRBO-1 to ensure that all data are modeled by a unified surrogate, which is essential for coherent noise allocation across the dataset.

4.4.2. Trust-Region-Guided Noise Injection

We now introduce a trust-region-guided noise injection (TR-NI) model that integrates spatial locality into the noise allocation process. The model receives the current trust region state (center $\boldsymbol{x}_c$ and length τ) from an external TuRBO opti-

mizer and assigns noise levels to each observation based on proximity to the trust region. Observations within the trust region $\mathcal{TR}$ are always assigned a minimal noise factor $\lambda_i = \epsilon_{\min}$ to ensure precise fitting in the current search area.

In its standard form, this method sets the noise factor for all observations outside the trust region to a fixed high value $\lambda_i = 1$. This static partitioning encodes a strong spatial prior by treating only the data near the current optimization region as trustworthy while systematically down-weighting distant points irrespective of their individual contribution to the model fit. Despite its simplicity, this variant effectively localizes model fidelity and reduces overfitting to outdated or irrelevant regions.

Algorithm 6 Trust-Region-Guided Noise Injection

Initialize TuRBO-style optimizer with internal trust region state
Initialize GP surrogate model with multiplicative Gaussian likelihood
for each Bayesian optimization iteration t **do**
    /* Optimizer Side */
    Evaluate new candidate and update trust region center $\boldsymbol{x}_c^{(t)}$ and length $\tau^{(t)}$
    /* Surrogate Model Side */
    Receive current trust region parameters $(\boldsymbol{x}_c^{(t)}, \tau^{(t)})$
    Compute noise factors $\boldsymbol{\lambda}$ based on proximity of each $\boldsymbol{x}_i$ to the trust region:

$$\lambda_i = \begin{cases} \epsilon_{\min} & \text{if } \boldsymbol{x}_i \in \mathcal{R}^{(t)} \\ 1 & \text{otherwise} \end{cases}$$

    Fit GP model using $\boldsymbol{\lambda}$ and update posterior
    Optimize acquisition function within trust region $\mathcal{TR}^{(t)}$ to propose next point
end for

The following two extensions introduce more adaptive strategies for allocating noise outside the trust region.

Binary Extension

The binary trust-region-guided noise injection (B-TR-NI) extension applies conditional noise adaptation exclusively outside the trust region. For each candidate point $\boldsymbol{x}_i \notin \mathcal{TR}^{(t)}$, the algorithm evaluates whether increasing λ_i leads to an improvement in the MLL. If this is the case, the noise factor is set to $\lambda_i = 1$. Otherwise, it remains

at $\lambda_i = \epsilon_{\min}$. This selective mechanism helps prevent the unnecessary discounting of informative observations that, while spatially distant, may still contribute meaningfully to the model fit.

Additive Extension

In the additive extension, noise is incrementally accumulated for each point that repeatedly falls outside the trust region. The update is governed by a user-defined step size $\Delta\lambda > 0$, which controls the rate at which noise increases over time. This scalar parameter determines the degree of which the model down-weights spatially irrelevant points across iterations. At each iteration, we update:

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \Delta\lambda \quad \text{if } \boldsymbol{x}_i \notin \mathcal{TR}^{(t)} \ . \tag{4.7}$$

Normalization can optionally rescale all λ_i values to lie within $[\epsilon_{\min}, 1]$, ensuring numerical stability and interpretability. However, such rescaling may distort the absolute differences between noise factors. For example, the sequences $[1, 2, 4, \epsilon_{\min}]$ and $[2, 4, 8, \epsilon_{\min}]$ yield the same normalized ratios, even though they represent different levels of historical irrelevance. This compression may bias the model against capturing fine-grained distinctions. Nevertheless, since λ_i values act only as relative weights on a globally learned noise scale σ_n^2, this distortion does not affect model expressivity in our setting. This variant, including normalization, is referred to as additive trust-region-guided noise injection (A-TR-NI). This approach encodes a memory of spatial irrelevance over time, gradually reducing the influence of consistently unhelpful data. Compared to the binary scheme, the additive model offers smoother noise evolution and better resilience against noise spikes due to one-off outlier detections.

4.4.3. Beam Search Trust-Region-Guided Noise Injection

To further increase the flexibility of trust-region-guided noise allocation, we again leverage BS [29] to explore multiple noise configurations in parallel. This method builds on the intuition that certain observations may consistently lie outside the current region of interest and should therefore be progressively down-weighted. Rather than committing to a single noise allocation strategy, BS maintains a population

of candidate noise vectors and iteratively refines them over time. We refer to this method as beam search trust-region-guided noise injection (BS-TR-NI).

At each iteration, the algorithm considers the current trust region $\mathcal{TR}^{(t)}$ and updates the beam by expanding each candidate configuration into two branches: (i) one in which all observations outside $\mathcal{TR}^{(t)}$ receive an additive noise increment $\Delta\lambda$, and (ii) one in which the noise allocation remains unchanged. Each resulting configuration is scored using the MLL, and the top-k candidates are retained for the next iteration. This process allows the model to simultaneously explore conservative and aggressive noise injection strategies and to accumulate noise selectively across iterations for persistently irrelevant points.

Algorithm 7 Trust-Region-Guided Beam Search Noise Injection

Initialize Beam $\mathcal{B}$ with initial noise vector $\boldsymbol{\lambda}^{(0)} = \epsilon_{\min} \cdot \mathbf{1}$ and associated MLL
for each Bayesian optimization iteration t **do**
    Receive trust region parameters $(\boldsymbol{x}_c^{(t)}, \tau^{(t)})$ from external optimizer
    Determine outside set $\mathcal{O}^{(t)} = \{i : \boldsymbol{x}_i \notin \mathcal{TR}^{(t)}\}$
    /* Beam Expansion */
    for each candidate $(\boldsymbol{\lambda}, \text{MLL}) \in \mathcal{B}$ **do**
        Branch 1 (Additive Update):
        Update: $\lambda_i \leftarrow \lambda_i + \Delta\lambda \quad \forall i \in \mathcal{O}^{(t)}$
        Normalize $\boldsymbol{\lambda}$ such that $\lambda_i \in [\epsilon_{\min}, 1]$ for all i
        Evaluate marginal log-likelihood: MLL_1
        Branch 2 (No Update):
        Retain $\boldsymbol{\lambda}$ unchanged
        Evaluate marginal log-likelihood: MLL_2
        Add both branches to candidate pool
    end for
    Select top-k candidates by MLL score to form next beam $\mathcal{B}^{(t+1)}$
    Fit GP using best candidate $\boldsymbol{\lambda}^*$ from $\mathcal{B}^{(t+1)}$
    Optimize acquisition function and evaluate new candidate
end for

Evaluating these trajectories against the MLL objective allows the method to implicitly learn which regions of the input space are consistently uninformative and should be down-weighted. Unlike B-TR-NI, BS-TR-NI maintains multiple hypotheses about which regions may be irrelevant, enabling the model to accumulate evidence over time before committing to noise injection decisions.

5. Empirical Results

This chapter presents empirical evaluations of the proposed noise injection strategies within the vanilla BO framework. All experiments use a GP surrogate model trained via MLL maximization and queried using analytical acquisition functions, specifically EI and UCB (with exploration parameter $\beta = 2$). The goal is to assess how targeted manipulation of observation noise affects optimization performance under model mismatch.

Across all experiments, the surrogate model is implemented as a `SingleTaskGP` with a zero mean function and a `ScaleKernel` composed with an `RBFKernel`, explicitly matching the definition given in (2.2). Baseline models employ a standard homoskedastic Gaussian likelihood, whereas the noise injection variants use the proposed multiplicative Gaussian likelihood from Section 4.1. All experiments are implemented in Python using a modular `Hydra` configuration [47], with `BoTorch` [4] and `GPyTorch` [11] serving as the primary modeling libraries.

Throughout all visualizations, shaded regions and error bars indicate the interquartile range, while lines and markers denote the median values across seeds.

Objective Functions

To evaluate the proposed methods, we benchmark them on a diverse set of synthetic objective functions widely used in the BO literature. These functions vary in modality, smoothness, and dimensional complexity to reflect a broad range of surrogate modeling challenges.

Each function is evaluated in one or more dimensional variants, ranging from low (2D) to very high-dimensional (100D) settings. This enables a systematic assessment of how model mismatch, noise injection interact, and the resulting BO performance changes across scales. All functions are posed as minimization tasks and affinely scaled to the unit hypercube $[0, 1]^d$. An overview of all benchmark functions is provided in Table 5.1.

Table 5.1.: Overview of benchmark objective functions.

Function	Dimensions	Landscape	Global Minimum
Branin [6]	2	3 minima, smooth	$f^\star = 0.398$
Hartmann [15]	3, 6	Narrow valleys	$f^\star = -3.86, -3.32$
Ackley [1]	2, 10, 20, 100	Flat, multimodal	$f^\star = 0.0$
Rastrigin [32]	2, 10, 20	Oscillatory, multimodal	$f^\star = 0.0$

Point of Transformation

A key distinction in the use of transformation strategies in BO lies in when they are applied. We distinguish between preprocessing transformations (PTs), which modify the observations prior to model fitting and acquisition optimization, and outcome transformations (OTs), which are applied internally during GP training and reversed before acquisition evaluation. PTs permanently alter the regression target, affecting all downstream components. Their use, as conceptually explored by Hutter *et al.* [16] for logarithmic transformations, primarily tests whether acquisition functions can exploit re-scaled representations to guide search more effectively. By contrast, OTs, which are available in BoTorch [4], affect only GP model fitting and are unwrapped before acquisition optimization, maintaining fidelity to the true objective. This distinction is visualized in Figure 5.1.

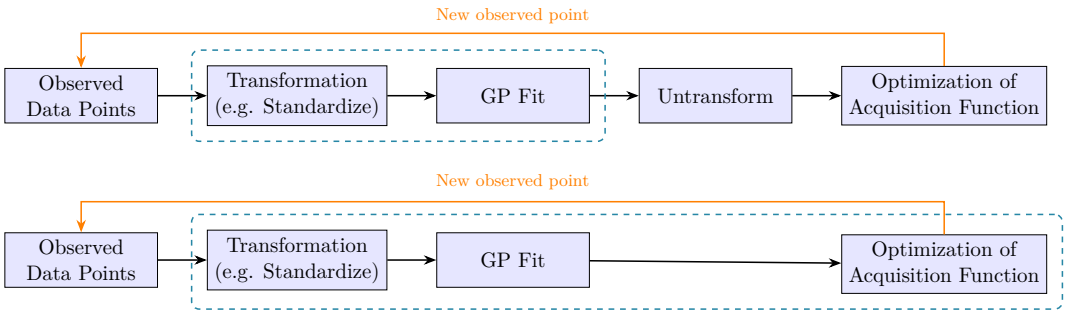

Figure 5.1.: Distinction between OT (top) and PT (bottom) workflows in BO. In OT, the transformation and its inverse are applied internally during model fitting and prediction, leaving the acquisition optimization in the original space. In PT, the data is transformed once as a preprocessing step, and model fitting and acquisition optimization occur entirely within the transformed space (turquoise dashed box).

All transformations are implemented as described in Section 2.3. For the log transform, we use $c = 1$ and shift the values $y = \log(y - \min(y) + 10^{-6})$ to also enable the transformation of negative values.

Benchmark Overview

The following sections present a structured empirical evaluation of the proposed methods. Benchmarks are grouped into three main parts, each corresponding to a distinct class of strategies: output transformations (Section 5.1), heuristic noise injection (Section 5.2), and trust region-guided noise injection (Section 5.3). Within each part, subvariants are evaluated individually to isolate the effects of specific design choices across acquisition functions and objective landscapes. A synthesis of overarching trends, limitations, and implications is provided in the Discussion (Section 5.4).

Unless stated otherwise, benchmarks are conducted using Sobol-initialized query points, fixed evaluation budgets, and no restarts. Initial length-scales are set to $\ell_0 = \sqrt{d}$, following the robust initialization strategy of Xu *et al.* [46]. Each experiment is repeated over 20 random seeds to assess robustness, except for sensitivity analyses, which are evaluated over 10 runs. Method configurations are named consistently throughout the results and follow the scheme introduced in Section 4. Names are extended with descriptors indicating applied transformations (e.g., Standardization-PT), initial noise configurations, and whether hyperparameters are refit during optimization. This ensures clear attribution of performance differences to specific modeling and algorithmic choices.

5.1. Output Transformation Benchmarking

To systematically quantify how data-side transformations mitigate model mismatch and influence BO performance, we benchmark standardization, logarithmic, and bilogarithmic output transformations, implemented either as OT or PT. This analysis allows us to disentangle the mechanisms that drive observed performance differences, such as changes in GP hyperparameter estimation and alterations to the optimization landscape. These results clarify the independent effects of output transformations and lay the groundwork for comparison with the noise injection methods

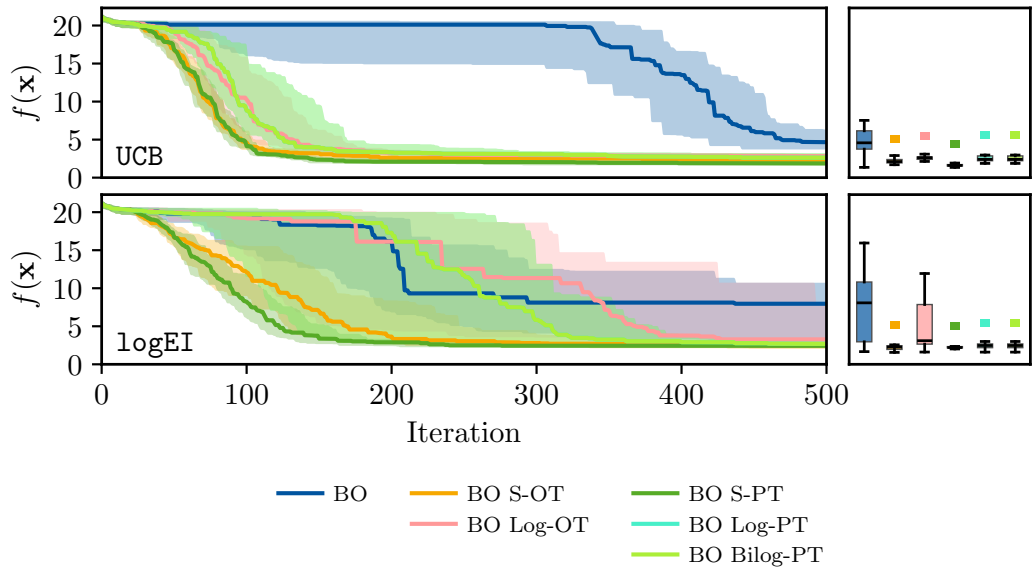

Figure 5.2.: Optimization performance of different output transformation strategies on the 10D Ackley function. The plots show the best-found function value over iterations. Box plots on the right summarize the distribution of final function values. Output transformations generally improved performance, with standardization showing the most consistent gains across acquisition functions, though differences between the points of transformation are present.

proposed later in this chapter.

BO without output transformations consistently underperformed compared to using transformed outputs. Specifically, Figure 5.2 illustrates slower convergence rates and earlier stagnation of optimization runs without transformations. This performance degradation can primarily be attributed to model mismatch, which is evidenced by distinct patterns in the learned GP hyperparameters (see Figure 5.3). Without transformation, the GP model exhibited an extremely small signal variance (σ_f^2), indicating insufficient capacity to capture variations in unscaled data. Additionally, the surrogate consistently learned excessively large length-scales (ℓ), reflecting an overly smooth representation of the objective landscape. Both phenomena contribute directly to poor surrogate model fit and, consequently, reduced optimization performance.

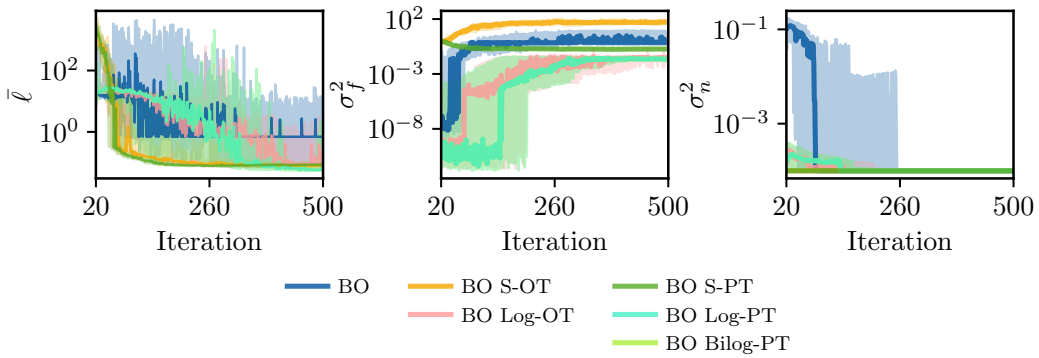

Figure 5.3.: Evolution of GP hyperparameters during optimization on the 10D Ackley function with LogEI. Output transformations led to more stable GP hyperparameter trajectories, with standardization especially reducing length-scale and noise variance fluctuations.

Standardization consistently improved BO performance relative to untransformed benchmarks across all considered scenarios. However, differences arose from the specific implementation of OT and PT, as evidenced in Figure 5.4. For instance, on the Ackley function benchmarks, standardization preprocessing transformation (S-PT) typically led to better convergence compared to OT, with the notable exception of the Ackley-100D setting under UCB. In contrast, logarithmic transformations generally underperformed relative to other methods, achieving comparable or slightly improved results only in isolated cases. No transformation type emerged as universally superior across all benchmarks, as performance depended strongly on both dimensionality and problem structure. However, in lower-dimensional settings, the performance differences between methods tended to be less pronounced. Due to the lack of a closed-form inverse, the bilogarithmic transformation could not be implemented as an OT variant with analytical acquisition functions. As a result, Bilog-OT was excluded from further empirical comparisons.

The choice of acquisition function further influenced performance and interacted notably with the selected output transformation. As illustrated in Figure 5.4, the same transformation could produce markedly different outcomes depending on whether UCB or LogEI was employed. This effect was particularly pronounced under logarithmic transformations. For example, Log-PT yielded strong performance on the Rastrigin-2D task when paired with LogEI, but its performance declined on

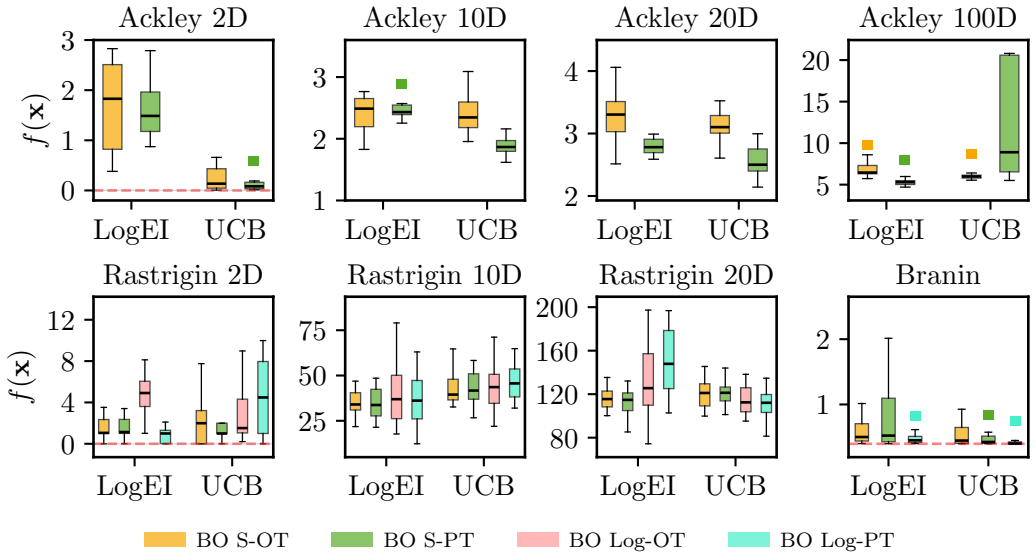

Figure 5.4.: Final optimization performance of output transformation strategies across multiple objective functions. Across many settings, PT achieved better results than OT, particularly when using standardization. For clarity, transformation variants with consistently poor performance are omitted.

Rastrigin-20D. Conversely, under UCB, Log-PT performed relatively better in the high-dimensional Rastrigin setting. These observations suggest that the effectiveness of logarithmic scaling is contingent not only on the problem landscape but also on its interaction with the acquisition strategy. More broadly, LogEI tended to yield superior performance on Rastrigin-type functions, whereas UCB showed advantages on Ackley-type landscapes. Importantly, the UCB acquisition function used across all benchmarks employed a fixed exploration parameter, suggesting that further tuning of this hyperparameter could yield additional improvements in performance.

Given the robust and consistently strong performance demonstrated by S-PT across various benchmark scenarios, subsequent empirical analyses within this thesis focus exclusively on this transformation strategy. Nevertheless, it is important to emphasize that other transformations may also yield beneficial effects in specific optimization contexts. Additional benchmark results can be found in A.3.

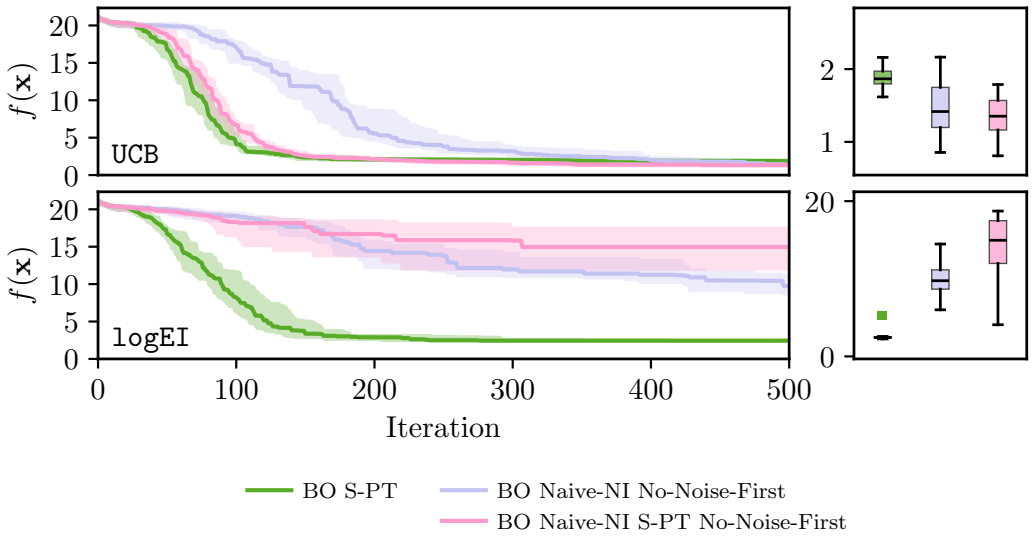

Figure 5.5.: Optimization performance of Naive-NI variations on the 10D Ackley benchmark. Naive-NI improved performance under UCB, and its combination with standardization further enhanced convergence speed and final outcomes, whereas no such benefit was observed under LogEI. The formatting is as in Figure 5.2.

5.2. Heuristic Noise Injection Benchmarking

Building on the strong S-PT baseline established in the previous section, we now assess heuristic noise injection methods that optimize the per-observation noise vector $\boldsymbol{\lambda}$. We begin with the simplest heuristic Naive-NI, and subsequently examine the performance of the remaining strategies introduced earlier, including a sensitivity analysis of their hyperparameter settings and the corresponding computational complexity.

The first benchmark evaluated the performance of Naive-NI relative to the established S-PT baseline. As shown in Figure 5.5, the plain Naive-NI strategy outperformed S-PT in terms of final objective values under the UCB acquisition function but required substantially more iterations to converge. Combining Naive-NI with standardization (*Naive-NI S-PT*) further enhanced performance by improving both convergence speed and final outcomes, although the convergence speed remained marginally slower than for the S-PT baseline. In contrast, no such bene-

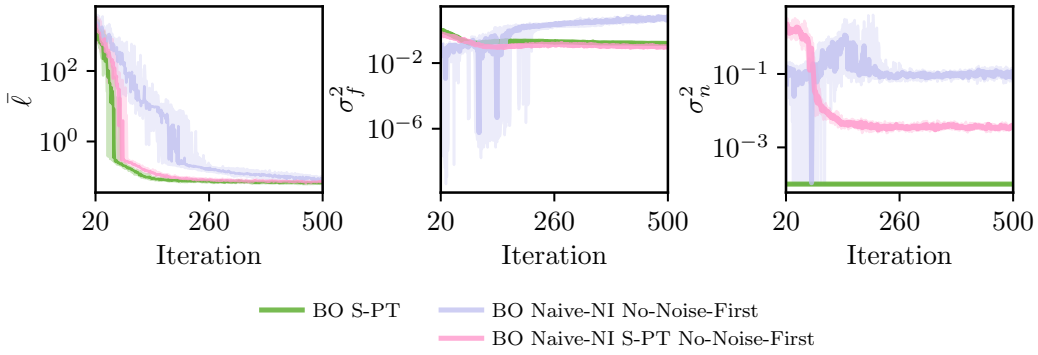

Figure 5.6.: Evolution of GP hyperparameters under Naive-NI variations on the 10D Ackley benchmark with UCB. Noise injection shifted the model's global estimate of σ_n^2 upward, because noise-injected points dominate the learning, while non-noise points do not constrain σ_n^2 to remain low. The formatting is as in Figure 5.3.

fit was observed under LogEI, where Naive-NI performed worse than the baseline, and standardization of Naive-NI even degraded results. These findings indicate that the utility of Naive-NI is acquisition-function dependent, with UCB benefiting more from the added local flexibility than LogEI.

Figure 5.6 illustrates the effect of noise injection on the evolution of GP hyperparameters. The previously observed large fluctuations in the length-scale trajectory without preprocessing are effectively reduced by all three configurations. Notably, the *Naive-NI S-PT No-Noise* variant converged to similar length-scale values as the S-PT baseline, while the plain *Naive-NI No-Noise* strategy converged more slowly. For the signal variance σ_f^2, both standardization and the combination with noise injection helped to stabilize the estimates. The noise variance panel shows that both noise injection variants learned markedly larger values of σ_n^2 than the S-PT baseline. The discrepancy reflects that noise-injected points dominate the MLL optimization due to their high variance, even though elevated uncertainty applies only to noise-injected observations. As a result, the GP overestimates σ_n^2, despite high uncertainty being confined to a subset of the data. Applying standardization before noise injection altered the temporal profile: the variance decreased during the first 50–100 iterations, then rose and remained more variable in later iterations, whereas the unstandardized variant reached an early peak and stabilized with lower variance. The

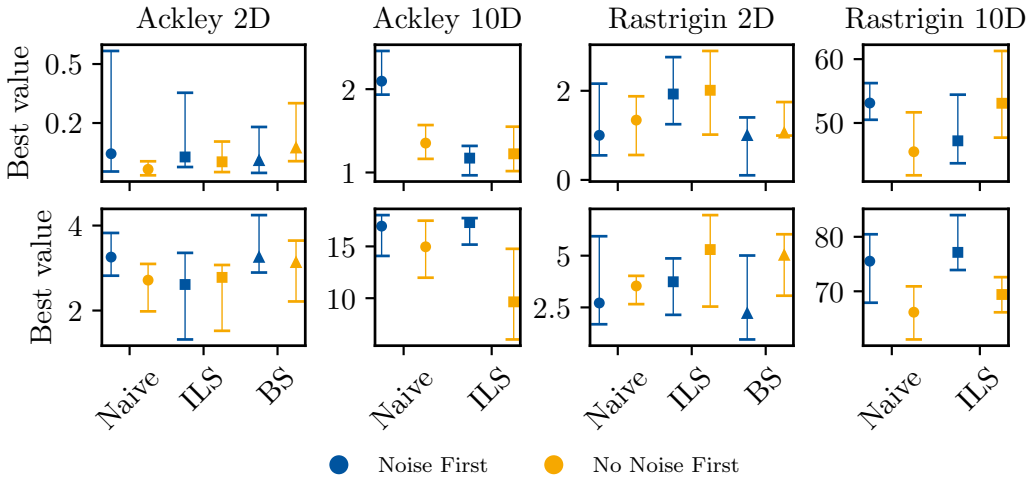

Figure 5.7.: Final optimization performance of heuristic noise injection methods under different initial noise configurations with UCB (top) and LogEI (bottom). The initial configuration influenced performance variably across methods and settings, with *No-Noise-First* generally yielding better results in higher dimensions, particularly for Naive-NI.

figure therefore indicates that preprocessing changes how the multiplicative likelihood allocates uncertainty, without affecting the overall trend of higher learned noise compared with the fixed-noise baseline.

Figure 5.7 shows that the initialization strategy used in heuristic noise injection with either *Noise-First* or *No-Noise-First* influenced the final performance, particularly for the naive strategy. Across benchmark functions, especially in higher-dimensional settings, *No-Noise-First* consistently outperformed *Noise-First*, indicating that it may serve as a more robust default strategy. This pattern suggests a preference for a more conservative, pointwise trust initialization, rather than imposing global mistrust from the outset and retracting it later. In contrast, for ILS-NI and BS-NI, results were less consistent and did not indicate a clear preference. This observation aligns with the broader sensitivity analyses of these methods discussed later in the chapter. All subsequent heuristic methods are initialized using the *No-Noise-First* strategy.

Refitting the GP after each update led to noticeable performance improvements in lower-dimensional settings, particularly when using the LogEI acquisition function

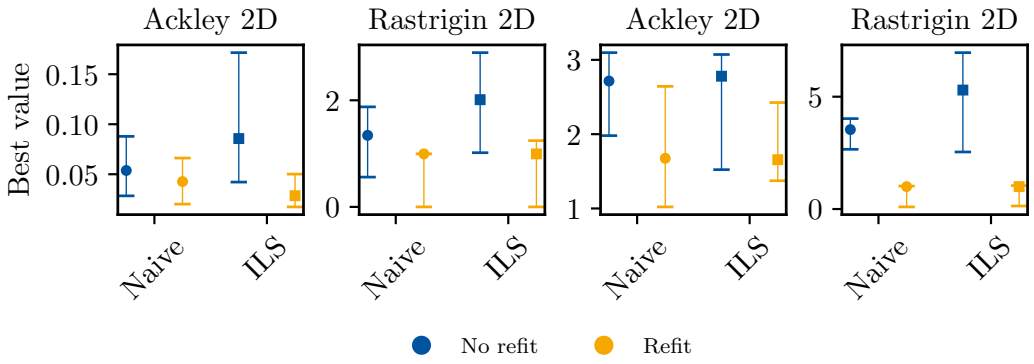

Figure 5.8.: Final optimization performance with and without GP hyperparameter refitting during heuristic noise injection. The left two columns show results under UCB, the right two under LogEI. Hyperparameter refitting consistently led to better performance across methods, objectives, and acquisition functions.

(see Figure 5.8). This effect was most pronounced on Ackley-2D and Rastrigin-2D, where refitting consistently reduced variance and improved final objective values across both noise injection strategies. For higher-dimensional tasks such as Ackley-10D or Rastrigin-10D, full refitting was not computationally feasible, as none of the runs completed within a 72-hour evaluation window. Given the substantial computational overhead introduced by frequent refitting, further discussed in Section 5.12, its benefits appear most relevant in low-dimensional regimes.

The sensitivity analysis of the perturbation size hyperparameter in ILS reveals largely stable performance across a wide range of values (Figure 5.9). Contrary to the initial hypothesis that an intermediate perturbation size might strike an optimal balance between local refinement and global exploration, final performance remains comparable even at the upper bound of the tested range. Notably, perturbation sizes approaching one, effectively resampling the noise vector in a manner similar to the Naive-NI strategy, do not result in significant performance degradation. These findings suggest that the ILS-NI procedure is relatively robust to perturbation magnitude, with no clear evidence for a distinct performance optimum. This raises the question of whether ILS-NI, in its current form, is capable of capturing more structured or dependent patterns in observation noise.

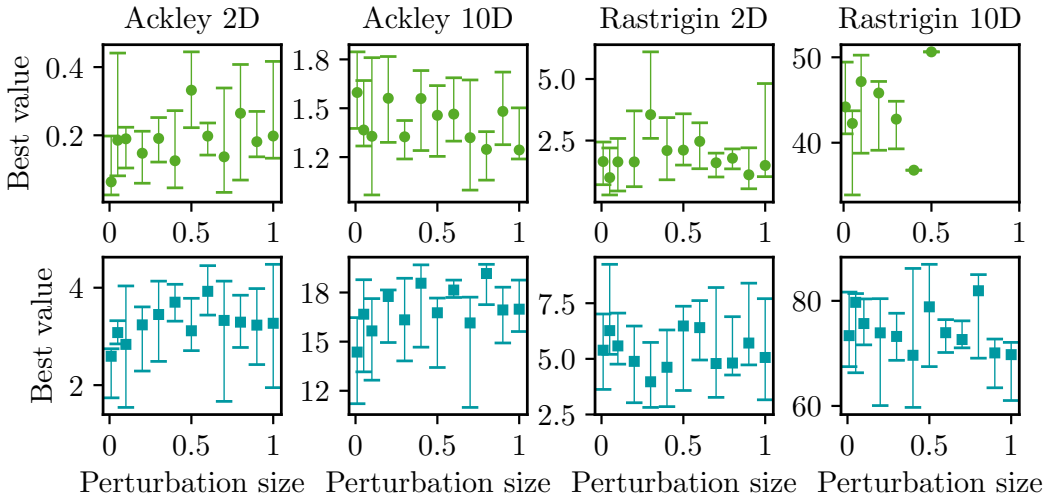

Figure 5.9.: Sensitivity analysis of ILS-NI to its perturbation size hyperparameter with UCB (top) and LogEI (bottom). ILS-NI showed stable performance across a wide range of perturbation sizes, indicating low sensitivity to this hyperparameter.

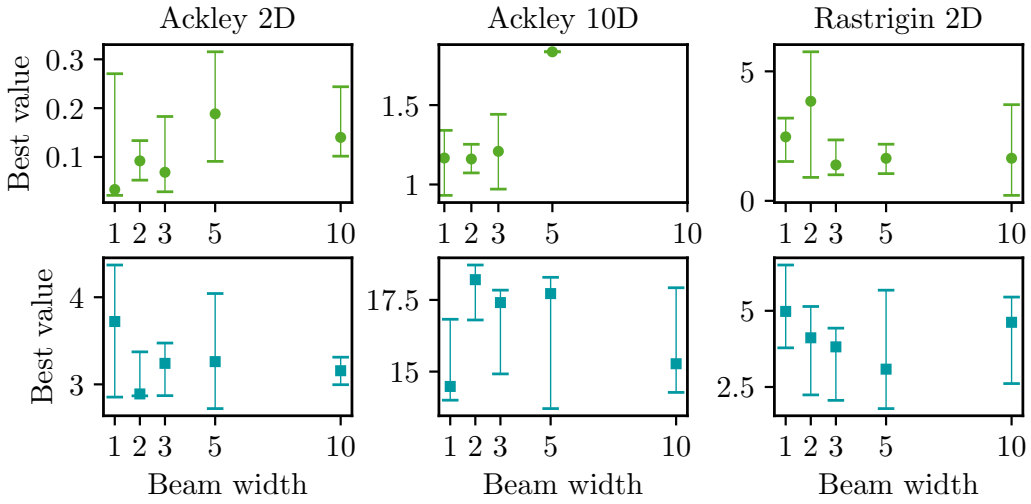

Figure 5.10.: Sensitivity analysis of BS-NI to its beam width hyperparameter with UCB (top) and LogEI (bottom). Performance did not consistently increase with beam width, indicating that broader candidate evaluation in BS-NI does not necessarily translate to better optimization outcomes.

A sensitivity analysis on the beam width parameter of the BS-NI strategy reveals no consistent improvement in performance with wider beams (see Figure 5.10). While wider beams were expected to enhance performance by preserving more candidate configurations per iteration and thereby capturing useful dependencies, the results do not support this hypothesis. Instead, they suggest that candidate configurations with higher MLL, identified through larger beams, do not reliably translate into improved optimization performance. Consequently, the assumption that maximizing MLL alone suffices to guide effective noise injection appears invalid in this context. Moreover, increasing the beam width substantially amplifies runtime, which, in conjunction with the absence of performance gains, further undermines the practical feasibility of BS-NI (Figure 5.12).

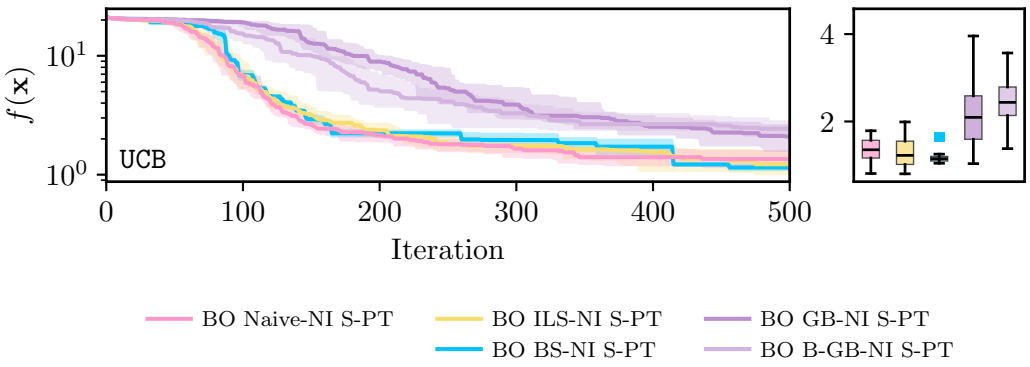

Figure 5.11.: Optimization performance comparison of heuristic noise injection methods on the 10D Ackley function. Naive-NI achieved competitive performance and was only marginally outperformed by the more complex methods ILS-NI and BS-NI in later iterations. The formatting is as in Figure 5.2.

A final performance comparison of all heuristic noise injection methods on the Ackley-10D task under the UCB acquisition function is presented in Figure 5.11. The gradient-based methods GB-NI and B-GB-NI failed to demonstrate improvements over the simpler Naive-NI strategy. ILS-NI achieved slightly better median performance in terms of final objective values, although this advantage only emerged in later iterations and was accompanied by a slower convergence speed. For BS-NI, only two random seeds successfully completed within a 72-hour benchmarking

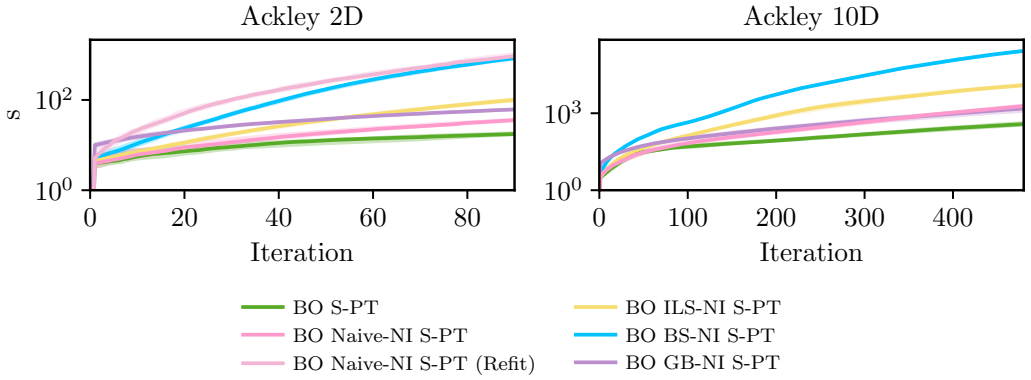

Figure 5.12.: Cumulative computational runtime (log-scaled) for different noise injection methods with UCB. Naive-NI added marginal runtime over standard BO, while BS-NI and refitted Naive-NI caused substantial overhead.

window, and the results presented are based exclusively on these runs. While suggestive, the limited sample precludes claims about robustness or reproducibility, and the narrow confidence intervals reflect sample size rather than stability. Since none of the methods demonstrated stable improvements under LogEI, only UCB is reported. Overall, Naive-NI emerges as the most promising approach when balancing optimization performance with computational efficiency, particularly given the marginal benefits observed from ILS-NI perturbation size analysis.

While runtime was not a primary evaluation criterion, Figure 5.12 compares the cumulative computational runtime of each method. As expected, more complex injection strategies such as ILS-NI and GB-NI incurred substantial runtime overhead, particularly in higher dimensions. By contrast, the Naive-NI added only modest overhead while maintaining competitive performance to ILS-NI and BS-NI.

5.3. Trust Region Noise Injection Benchmarking

After exploring heuristic noise injection within a standard BO setup, we now shift our focus to algorithmic adaptations of the BO framework itself, specifically examining the high-dimensional optimization variant TuRBO. Our results replicate prior findings showing that TuRBO significantly improved convergence in high-dimensional

settings [8], [33], [34]. We can also confirm that applying standardization within this framework further boosted performance. Given the model-agnostic nature of our noise injection approach, we next evaluate its effectiveness when integrated into the TuRBO paradigm to enable localized control over model trust.

Figure 5.13 summarizes the performance of plain TuRBO, its S-PT variant, and several noise injection extensions. In this paragraph, we first focus on the integration of Naive-NI into the TuRBO framework. Contrary to its moderate success in the vanilla BO setup, Naive-NI consistently degraded performance when combined with TuRBO, under both UCB and LogEI. This underperformance indicates that uninformed, global noise assignment may interfere with the spatial modeling assumptions of trust-region methods.

To overcome the limitations of globally applied heuristic noise, we next examine trust-region-aware variants that adapt noise placement based on the spatial locality of the optimization process. As shown in Figure 5.13, injecting noise selectively outside the current trust region with the TR-NI variant leads to consistent performance improvements under both acquisition functions. In contrast, the B-TR-NI variant, which applies noise only when the overall MLL increases, performed worse. These findings reinforce the conclusion from the heuristic benchmarks that optimizing MLL on a pointwise basis does not reliably translate into better optimization outcomes. Instead, applying noise consistently based on spatial structure, by down-weighting observations in less-explored regions, proved to be more effective in mitigating model mismatch and improving the surrogate model's fit.

The A-TR-NI variant achieved similar performance to TR-NI, but with slightly faster convergence and improved final values under UCB. This improvement likely stems from A-TR-NI's ability to incrementally accumulate noise for observations that persistently lie outside the trust region, enabling the model to adaptively de-emphasize spatially uninformative regions over time. The increment parameter $\Delta\lambda$ was fixed to one in all experiments but constitutes an additional degree of freedom that could be tuned to adapt the method to specific problem settings. Importantly, normalization proved essential for preserving numerical stability: without it, noise values grew too large and disrupted the GP fitting, particularly affecting noise variance estimation.

BS-TR-NI, when combined with additive noise injection outside the trust region, failed to yield meaningful performance improvements and, in some cases, performed

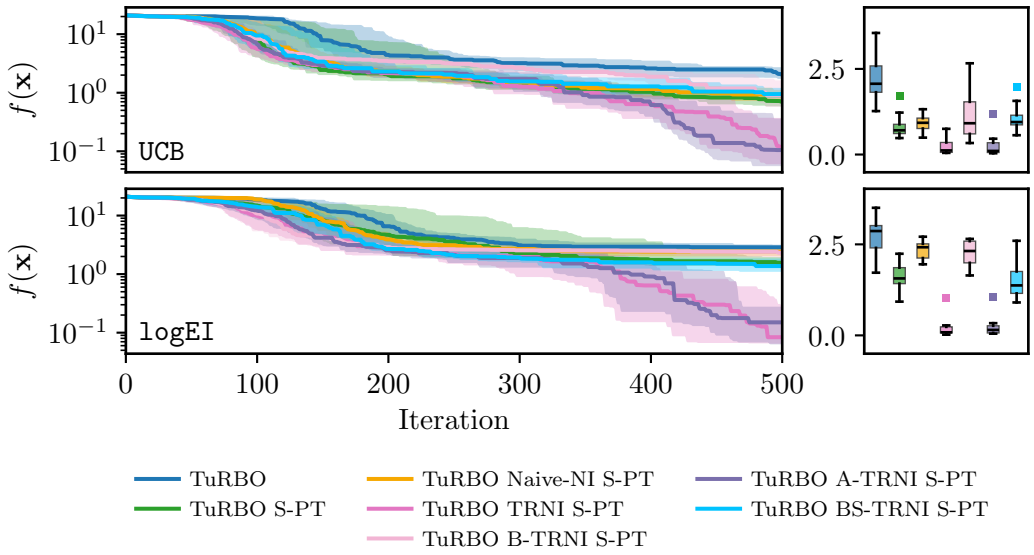

Figure 5.13.: Optimization performance of TR-NI variants within the TuRBO framework on the 10D Ackley function. Naive-NI degraded TuRBO performance, while consistently injecting noise outside trust regions improved results, with A-TR-NI emerging as a promising extension. The formatting is as in Figure 5.2.

worse than simpler alternatives. This result mirrors findings from the heuristic benchmarks, where BS-NI also lagged behind. Although BS-TR-NI is designed to preserve diverse candidate solutions and promote modeling flexibility, its reliance on maximizing MLL does not consistently lead to better optimization performance. These results further support the conclusion that structure-aware noise placement outperforms purely metric-driven strategies.

Importantly, the integration of noise injection into TuRBO did not lead to a substantial increase in runtime. Despite the added complexity of dynamically updating noise factors within each local region, the overall computational cost remained comparable to that of standard TuRBO runs. As expected, BS-TR-NI showed a moderate runtime increase due to beam expansion, but remained within a feasible range relative to BS-NI. Detailed runtime comparisons for all trust region variants are provided in Appendix A.3.

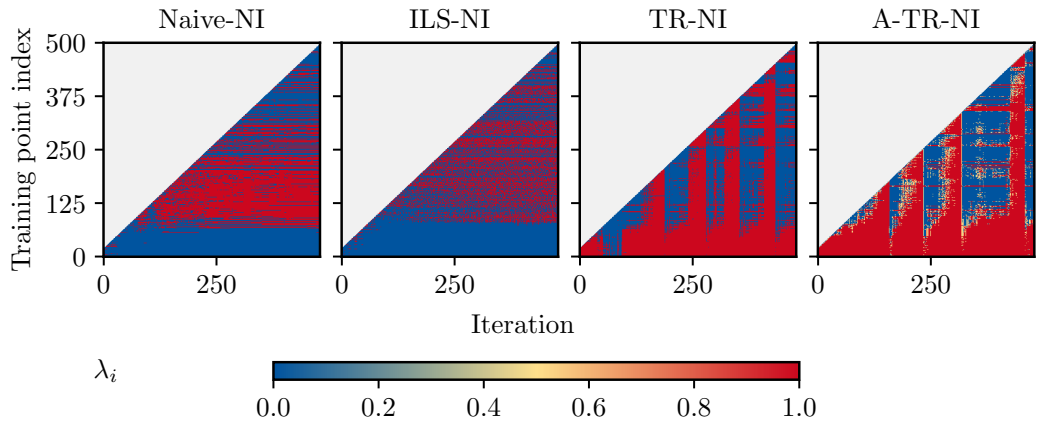

Figure 5.14.: Temporal evolution of the noise vector $\boldsymbol{\lambda}$ for different injection strategies on the 10D Ackley benchmark with UCB. Each heatmap visualizes the noise value λ_i assigned to each training point (y-axis) at each BO iteration (x-axis). Color encodes the noise value, where blue indicates high trust ($\lambda_i \to 0$, minimal noise) and red indicates low trust ($\lambda_i = 1$, maximal noise).

Noise Injection Pattern

In addition to evaluating optimization performance, we now examine the spatial and temporal patterns of noise allocation across iterations. This analysis provides insight into the internal mechanisms of the different noise injection strategies.

Figure 5.14 visualizes the evolution of the injected noise vector $\boldsymbol{\lambda}$ for all training points across optimization iterations. The first two panels correspond to the Naive-NI and ILS-NI heuristic methods, both of which operate without trust region constraints. These strategies produce highly structured horizontal banding patterns, indicating that specific training points are consistently down-weighted regardless of the current optimization phase. Remarkably, early observations (low indices) are assigned minimal noise throughout, as seen in the persistent blue block at the bottom of each panel. This static treatment suggests these points act as anchors for the surrogate model, rather than being actively re-evaluated for relevance. Compared to the naive baseline, the ILS-NI variant resulted in a more scattered horizontal pattern of injections, with more training points intermittently down-weighted. This likely stems from ILS's perturbation-based exploration.

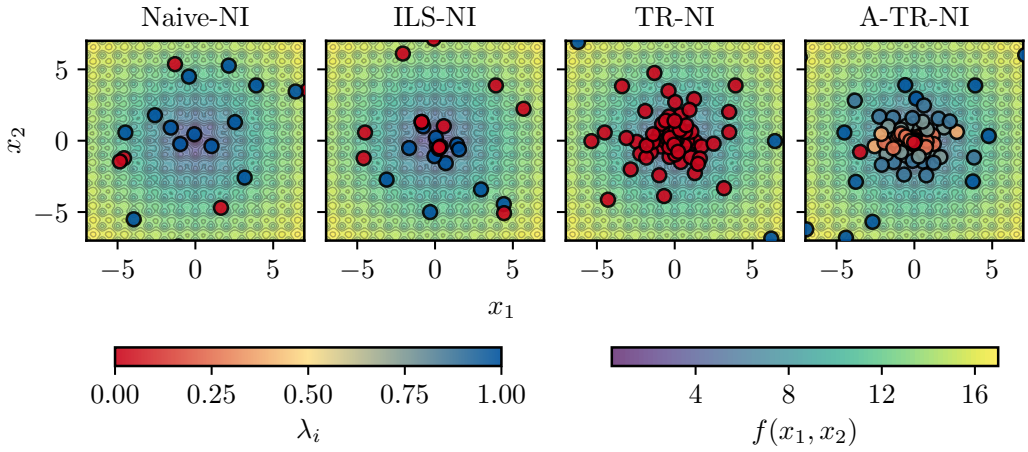

Figure 5.15.: Spatial distribution of final injected noise values on the 2D Ackley objective landscape with UCB. Each evaluated point from a representative BO run is colored according to its final noise level λ_i. Red indicates high trust (low noise), while points with higher injected noise trend towards blue. The results for four different noise injection strategies are shown against the objective function's contours.

In contrast, the trust-region-based methods in the third and fourth panels inject noise based on a training point's spatial relation to the current trust region. This spatial conditioning leads to vertically structured patterns, where noise is selectively assigned to points located outside the active region at each iteration. The third panel reflects the forced injection variant TR-NI, which applies either full noise or minimal noise depending on the location, resulting in sharp red-blue segmentation across iterations. In contrast, the additive variant A-TR-NI shown in the fourth panel accumulates noise over time, yielding smoother λ_i gradients that span the full $[\epsilon_{\min}, 1]$ range. This gradual adjustment produces more diffuse transitions, reflecting the model's ability to encode temporal irrelevance through incremental down-weighting.

Figure 5.15 visualizes the spatial distribution of final noise allocations across the 2D Ackley landscape. Each evaluated point is colored by its final noise value λ_i, where blue indicates high trust (low noise) and red denotes down-weighting through noise injection. The first two panels (Naive-NI and ILS-NI) illustrate how heuristic strategies operate without spatial priors. ILS-NI exhibits a relatively diverse trust pattern due to its perturbation-based updates, distributed across the whole search

space. In contrast, Naive-NI assigns low noise to only a few scattered points, which are not concentrated near the global optimum, suggesting that it lacks spatial selectivity and injects noise more uniformly across the search space. In contrast, the trust-region-based strategies (TR-NI and A-TR-NI) reveal clear spatial structure: TR-NI applies binary assignments based solely on trust region inclusion, resulting in a sharp segmentation between trusted and down-weighted points. The additive variant A-TR-NI accumulates noise over time, producing smoother gradients across the entire input space. This soft adjustment enables the model to encode temporal irrelevance and to gradually reduce the influence of consistently uninformative points. These findings illustrate how trust-region-guided strategies embed a spatial inductive bias that prioritizes local fidelity, whereas heuristic methods allow for more globally adaptive trust patterns. While the figure illustrates only a single optimization run and shows final noise values rather than their full trajectories, it nonetheless highlights distinct trust allocation patterns that align with each method's underlying selection mechanism.

5.4. Discussion

Empirically, we showed that targeted noise injection is able to adaptively mitigate model mismatch in BO by reducing the influence of irrelevant and misleading observations. Instead of regarding the uncertainty to be constant, the proposed method uses adaptive noise allocation to regularize the surrogate behavior, so that the model predictions become better aligned with the true objective function in the important areas of the search space.

A key insight is that both output transformations and noise injection suppress overconfidence in regions with poor model fit. While transformations act globally, noise injection enables localized uncertainty modulation without altering the GP structure. This selective flexibility helps explain why even simple strategies like Naive-NI, particularly when combined with standardization, consistently performed better than traditional methods across benchmarks.

Across both heuristic and trust region settings, the results consistently show that MLL maximization alone is not a reliable guide for improving BO performance under model mismatch. While Naive-NI applies a single-step MLL optimization and yields strong results, more elaborate strategies like ILS-NI and BS-NI, which iteratively

maximize MLL, often perform worse despite achieving higher likelihoods. Similarly, in the trust region framework, gating decisions based on MLL thresholds underperform compared to spatially informed additive schemes. These findings suggest that optimizing the surrogate model's fit, as measured by MLL, does not necessarily align with better acquisition outcomes. Notably, refitting GP hyperparameters during noise optimization improved performance in low-dimensional settings, but offered limited benefit in higher dimensions while substantially increasing runtime. This highlights that surrogate adaptivity can be helpful when sample complexity is manageable, but must be balanced against computational constraints.

The integration of noise injection into the TuRBO framework demonstrates that uncertainty shaping remains effective across diverse BO settings. More importantly, the success of A-TR-NI reveals a broader modeling principle: trust in data should be dynamically adapted based on optimization context rather than inferred solely from MLL. By conditioning noise on spatial relevance, such as whether points lie within an active trust region, TR-NI implements a form of soft forgetting that attenuates outdated or misleading observations without discarding them. Our results show that context-aware noise modulation, as implemented by TR-NI and A-TR-NI, improves performance by prioritizing surrogate fidelity in regions actively explored during optimization. This contrasts with likelihood-driven calibration, which can overfit to globally irrelevant points and underperform, especially in scenarios where high dimensionality exacerbates model mismatch.

Taken together, these findings position targeted noise injection as a principled approach for improving surrogate model reliability under conditions of mismatch. By treating observation noise as a tunable modeling variable, rather than a fixed nuisance, BO can adapt more flexibly to structural discrepancies and data irregularities. The approach's compatibility with diverse acquisition functions and frameworks, including TuRBO, highlights its potential as a broadly applicable mechanism for uncertainty shaping.

6. Conclusion and Outlook

This thesis addressed the challenge of model mismatch in BO, where violated surrogate assumptions can harm optimization performance. To mitigate this, we introduced targeted noise injection as a data-side adaptation that selectively down-weights misleading or outdated observations through per-point noise scaling. This approach enhances the adaptability of the observation model within heteroskedastic noise settings. In trust-region-guided variants, the method additionally incorporates spatial relevance, directing modeling capacity toward regions of the input space where optimization is actively progressing.

Initial benchmarks on standard output transformations illustrated the general mechanism by which data-side adaptations can influence GP hyperparameter learning to reduce model mismatch. However, these global transformations apply uniformly and lack the flexibility to account for localized outliers. In contrast, targeted noise injection enabled a more individualized adjustment of the influence of individual observations. Empirically, it produced comparable effects on GP hyperparameter learning while offering greater adaptability. Among these methods, the lightweight Naive-NI variant, particularly when combined with standardized data, consistently improved optimization performance compared to global output transformations. Building on this, the TR-NI framework enabled targeted hyperparameter learning by concentrating modeling capacity on relevant regions of the search space. Finally, the additive variant A-TR-NI introduced a memory-based attenuation mechanism that progressively down-weighted persistently irrelevant regions. Across benchmarks, this informed approach achieved the strongest overall performance.

A central insight from our evaluation is that MLL, while broadly effective for fitting global GP hyperparameters, may be insufficient when used to optimize per-point noise levels under model mismatch. In particular, heuristic noise injection methods involving iterative MLL maximization frequently underperformed relative

to simpler approaches like Naive-NI. This disconnect suggests that when applied to flexible noise models, such as our multiplicative Gaussian likelihood, MLL may overfit the observation-level noise structure without improving the surrogate model's predictive utility for acquisition.

These findings prompt future research toward alternative noise optimization criteria that better capture predictive utility. One promising direction is leave-one-out negative log predictive density (LOO-NLPD) [40], which directly evaluates generalization to unseen data and may be more aligned with acquisition performance than MLL. Further, the additive trust region noise injection (A-TR-NI) strategy could be extended by dynamically adapting the step size $\Delta\lambda$ based on trust region shifts across iterations, allowing for more context-aware noise updates. Finally, applying these methods in real-world optimization problems with structured noise, constraints, or nonstationarities would be essential to assess their practical robustness.

A. Appendix

A.1. Algorithmic Details for Gradient-Based Noise Injection

This appendix documents the key implementation details used for the gradient-based noise injection baselines described in Section 4.2.

Continuous Factor Optimization

Initialization. Noise factors λ_i are drawn from $\mathcal{N}(0.5, 0.05^2)$ and then clamped to $[\epsilon_{\min}, 1]$. This ensures feasible starting points while introducing minor random asymmetry across data points.
Optimizer. The Adam optimizer is used with a dimension-adaptive learning rate as specified in Appendix B.
Gradient Clipping. Each update clips ℓ_2 gradient norms to $1/\max(1, d/5)$ to maintain stability.
Early Stopping. Optimization terminates when the MLL fails to improve for p consecutive iterations, as determined by plateau detection, where p is the patience.

Binary Factor Optimization

The binary variant uses the same optimizer, clipping, and stopping logic as specified above in the continuous factor optimization.
Thresholding. After convergence, binary decisions are made via $\lambda_i = 1$ if $\sigma(\eta_i) > 0.5$ and $\lambda_i = \epsilon_{\min}$ otherwise.

A.2. Length-Scale Weighted Trust Region Rescaling

To account for anisotropic sensitivity in the input space, the trust region in TuRBO [8] is rescaled at each iteration using the learned length-scales of the surrogate model.

This rescaling gives larger side lengths to dimensions with larger learned length-scales (smoother variation) and smaller side lengths to dimensions with smaller length-scales (more rapid variation).

Following the original TuRBO implementation, a per-dimension weight vector $\boldsymbol{w} \in \mathbb{R}^d$ is computed by normalizing the learned length-scales $\boldsymbol{\ell} \in \mathbb{R}^d$ as:

$$w_i \;=\; \frac{\ell_i}{\left(\prod_{j=1}^{d} \ell_j\right)^{1/d}}, \qquad i = 1, \ldots, d. \tag{A.1}$$

This normalization ensures that the trust region retains a constant volume across iterations, while its shape adapts to directional sensitivities. Specifically, the weight vector $\boldsymbol{w}$ is re-scaled to have a geometric mean of one:

$$\left(\prod_{i=1}^{d} w_i\right)^{1/d} = 1.$$

The trust region bounds are then constructed as:

$$\text{bounds} = \left[\mathbf{x}_c - \boldsymbol{w} \cdot \frac{\tau}{2},\, \mathbf{x}_c + \boldsymbol{w} \cdot \frac{\tau}{2}\right] \cap [0,1]^d \tag{A.2}$$

Here, $\mathbf{x}_c \in \mathbb{R}^d$ denotes the trust region center, typically the best observed point, and τ is the global trust region width. The intersection with the unit hypercube ensures that the rescaled bounds remain within the normalized input domain.

A.3. Additional Empirical Results

Exploration–Exploitation Behavior

To complement the main results, we briefly examine the impact of data-side adaptations on the exploration–exploitation trade-off during BO. While not directly related to the goal of mitigating model mismatch, these behaviors offer additional insights when interpreting optimization performance.

To quantify exploration behavior, we use an average minimum distance metric that measures how far each newly proposed query point is from the existing training set. Let $\mathcal{D}_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ denote the normalized training inputs at iteration t, and

let $\mathcal{X}_{t+1}$ be the newly proposed points. The exploration metric is defined as:

$$E_{t+1} = \frac{1}{|\mathcal{X}_{t+1}|} \sum_{\mathbf{x} \in \mathcal{X}_{t+1}} \min_{\mathbf{x}' \in \mathcal{D}_t} \|\mathbf{x} - \mathbf{x}'\|_2$$

This represents the average minimum Euclidean distance between the new candidates and all existing points.

Figures A.1, A.3, and A.2 show the evolution of this metric for output transformations, heuristic noise injection, and trust region noise injection, respectively. While we do not draw causal conclusions, visible differences in exploration behavior suggest that targeted noise injection may influence the spatial distribution of proposed points. Such effects should be considered when interpreting improvements in BO performance.

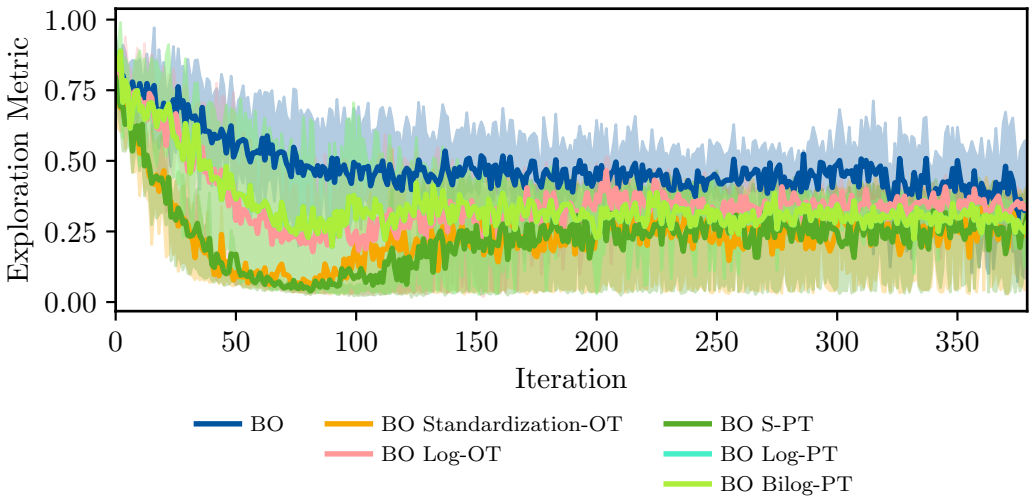

Figure A.1.: Exploration behavior, measured by average minimum distance, for various outcome transformations on the 10-dimensional Ackley benchmark with UCB.

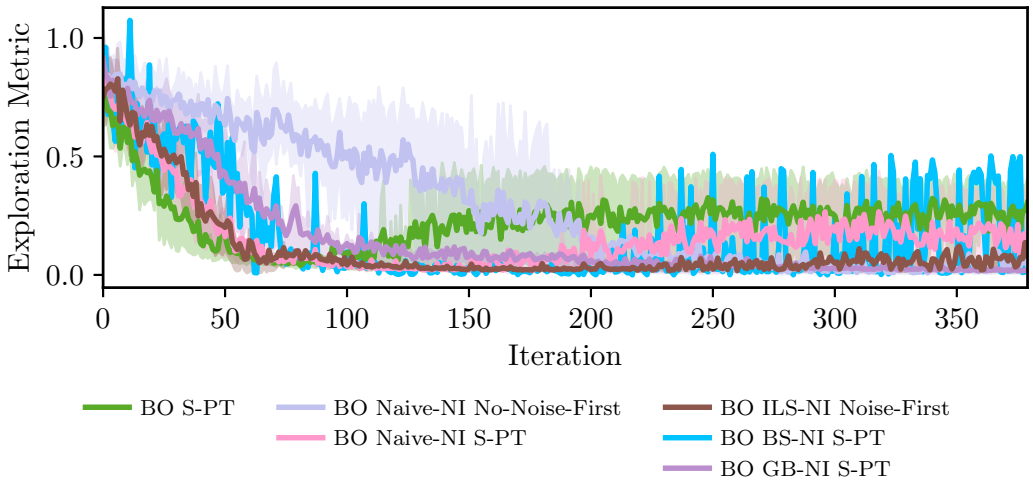

Figure A.2.: Exploration behavior, measured by average minimum distance, for different heuristic noise injection strategies on the 10-dimensional Ackley benchmark with UCB.

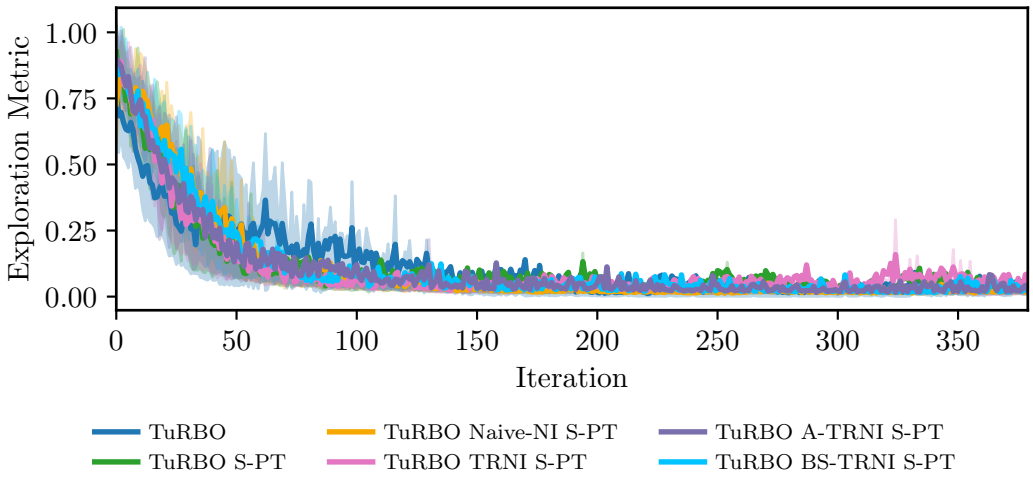

Figure A.3.: Exploration behavior, measured by average minimum distance, for TR-NI variants on the 10-dimensional Ackley benchmark with UCB.

Transformation Benchmarking

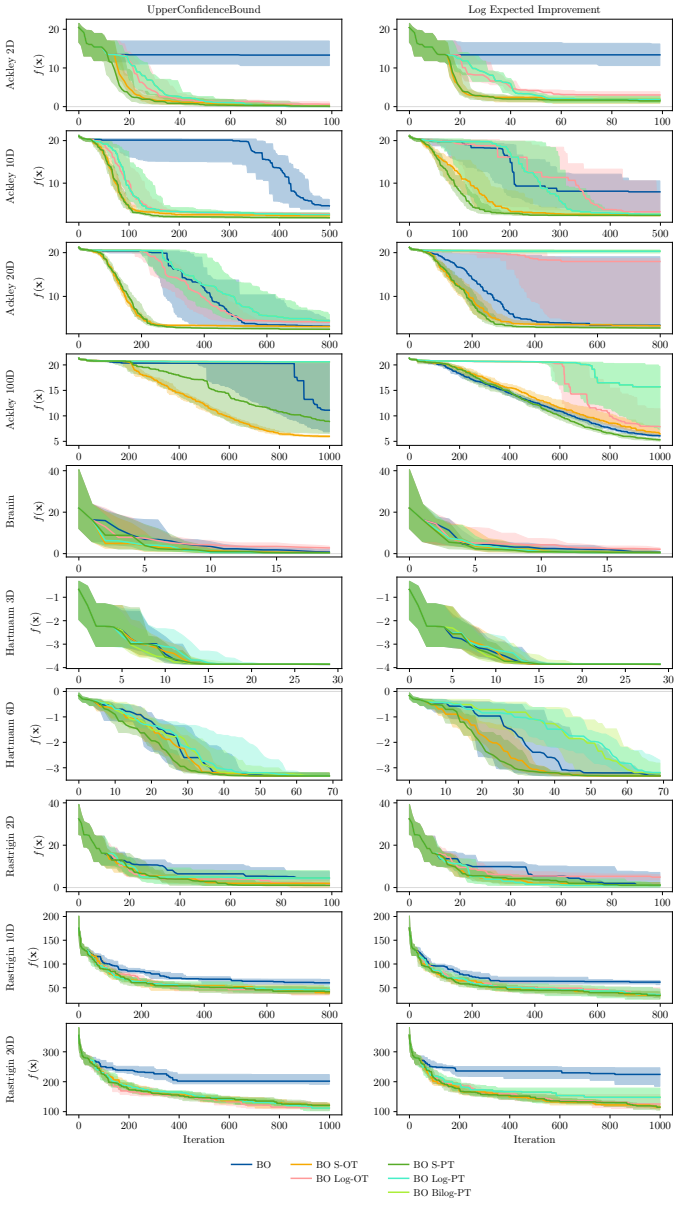

Figure A.4.: Regret trajectories for output transformation benchmarking across all evaluated objective functions and dimensional settings.

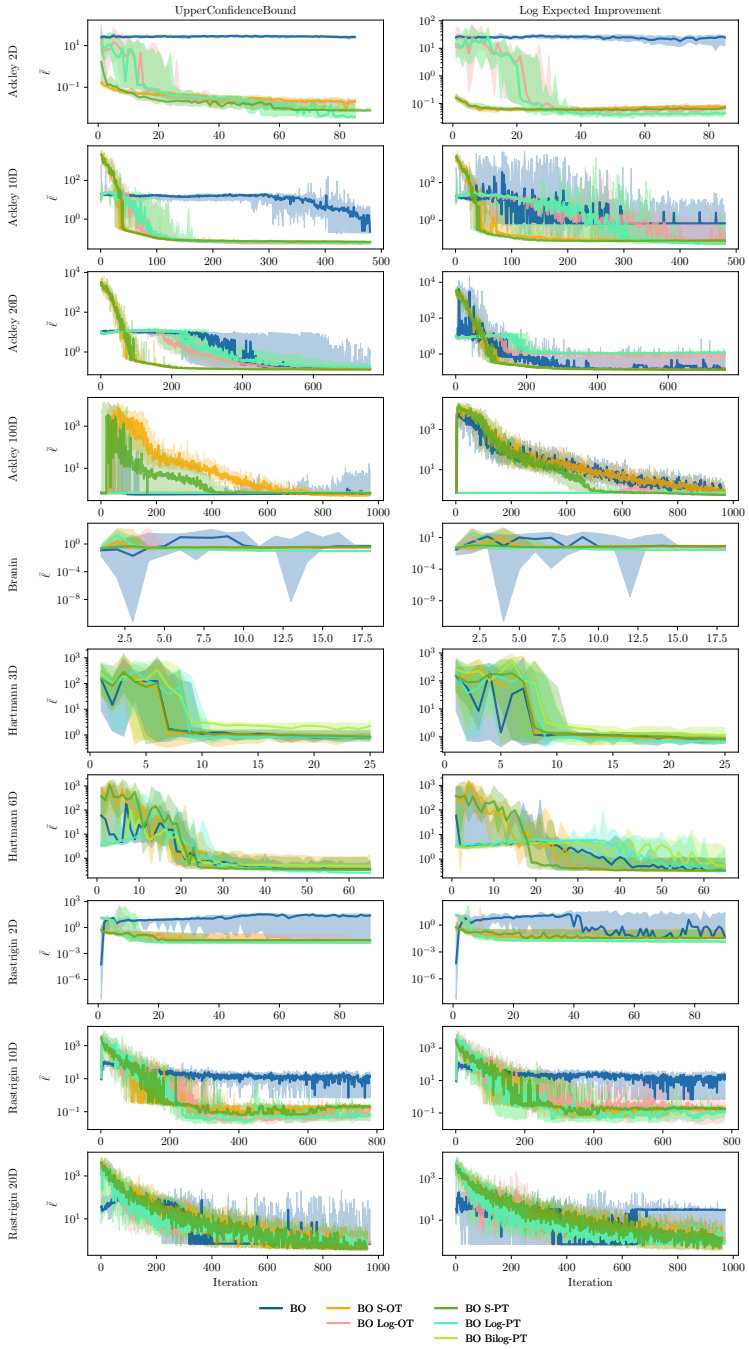

Figure A.5.: Evolution of the mean length-scale hyperparameter during the output transformation benchmarking across all tested scenarios.

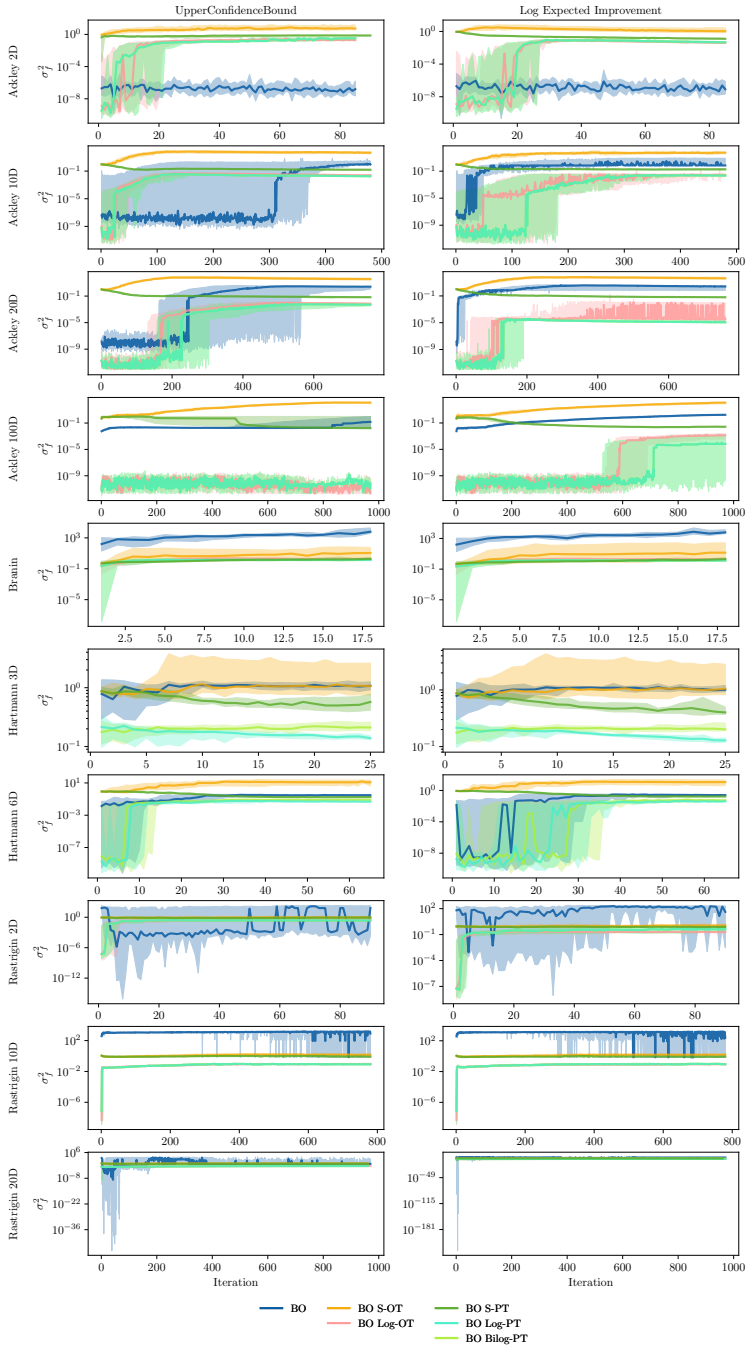

Figure A.6.: Evolution of the signal variance hyperparameter during the output transformation benchmarking across all tested scenarios

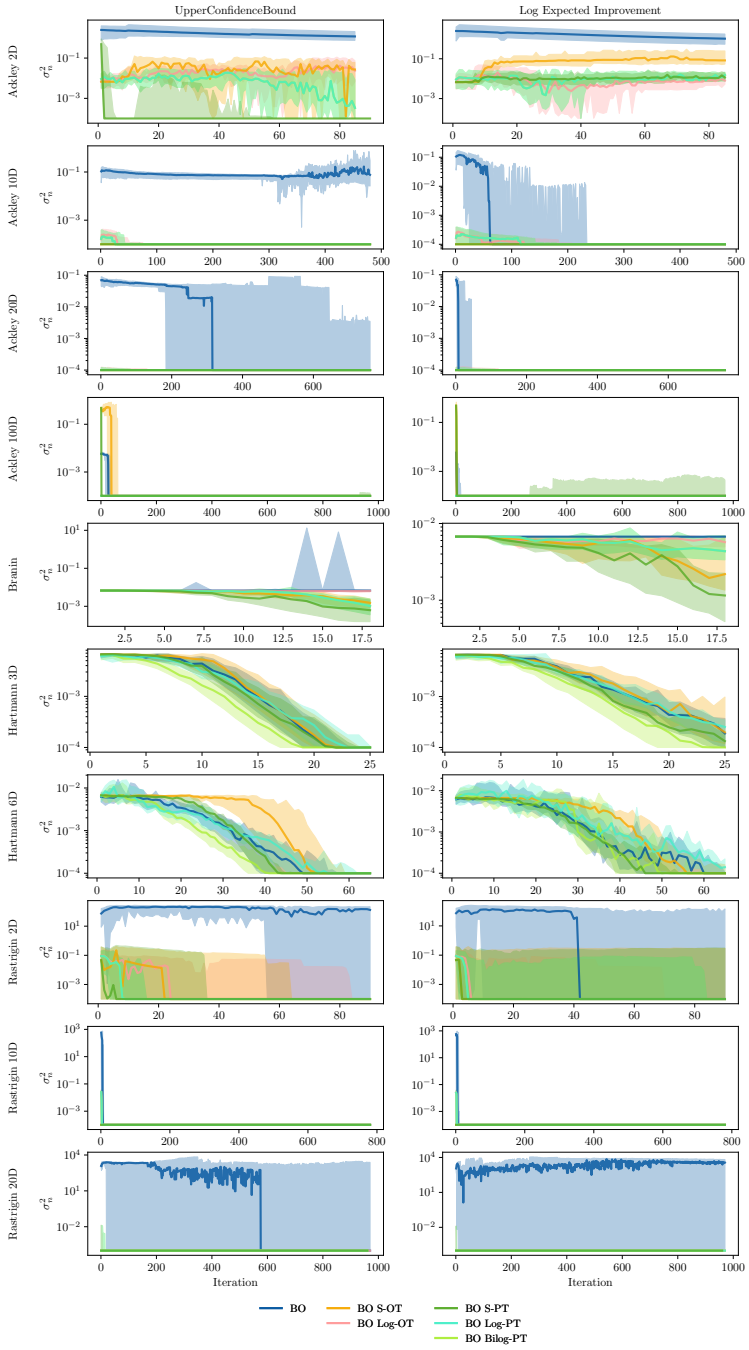

Figure A.7.: Evolution of the noise variance hyperparameter during the output transformation benchmarking across all tested scenarios.

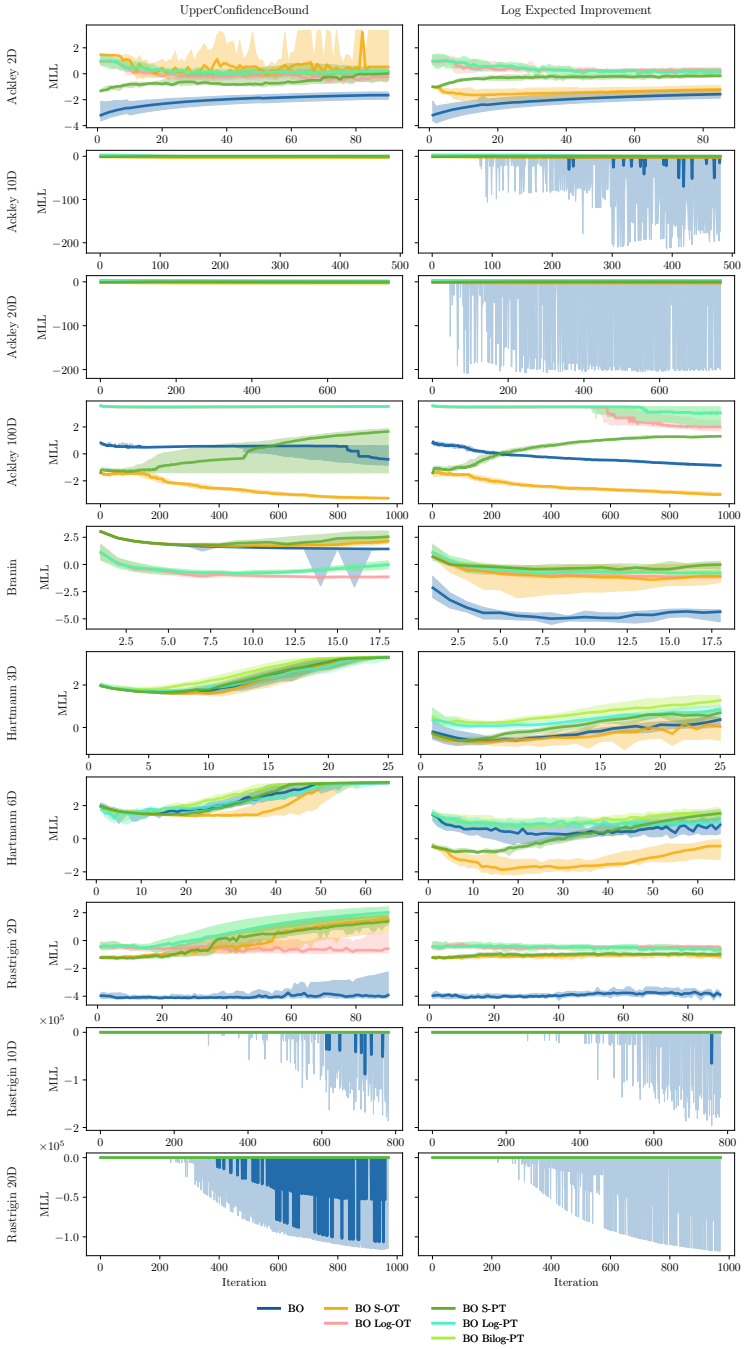

Figure A.8.: Evolution of the MLL during the Output Transformation benchmarking across all tested scenarios.

Heuristic Noise Injection Benchmarking

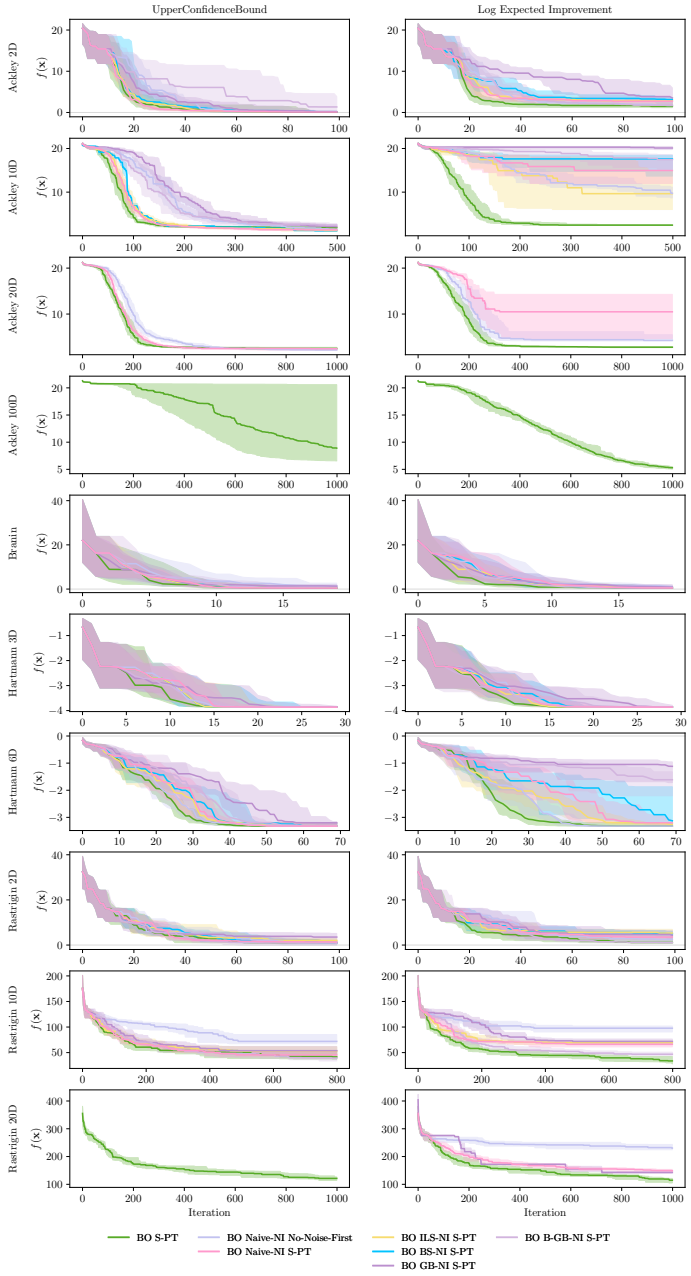

Figure A.9.: Regret trajectories for the heuristic noise injection benchmarking across all tested objective functions and dimensionalities.

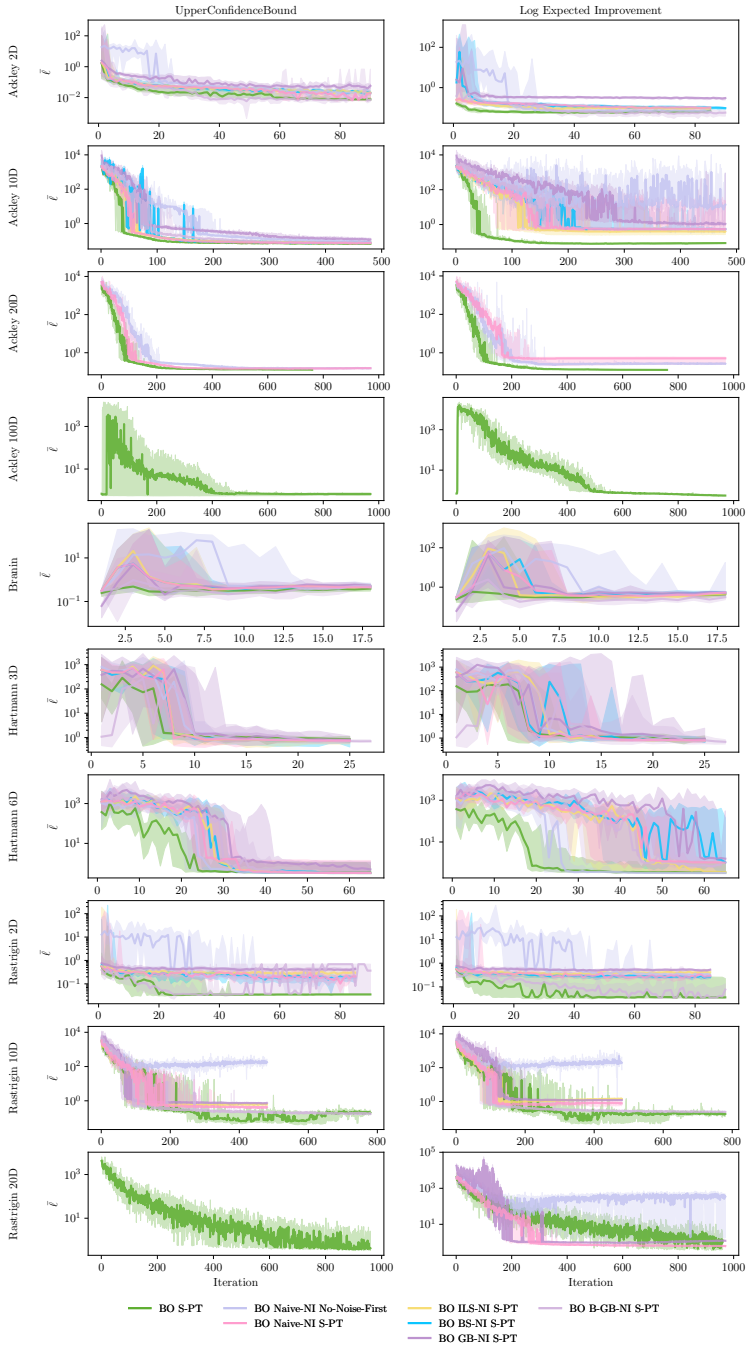

Figure A.10.: Evolution of the mean length-scale hyperparameter during the heuristic noise injection benchmarking across all tested scenarios.

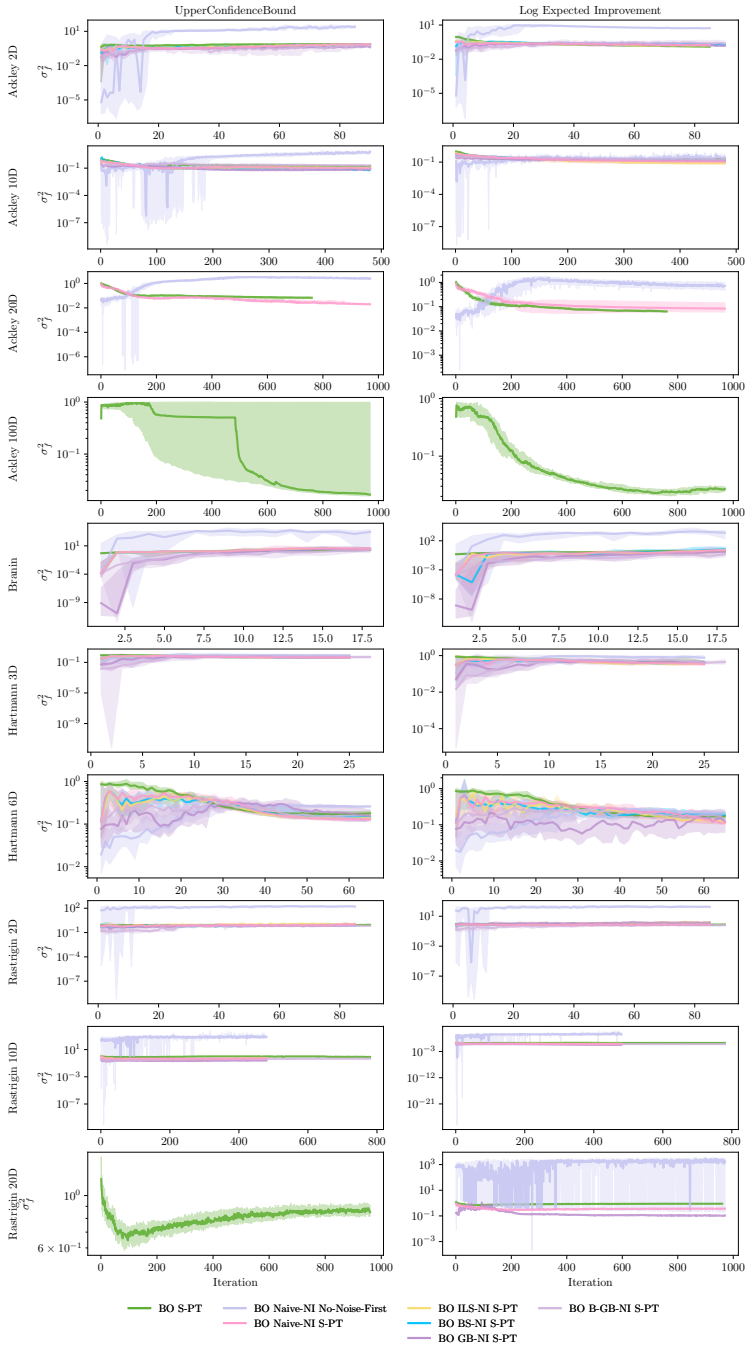

Figure A.11.: Evolution of the signal variance hyperparameter during the heuristic noise injection benchmarking across all tested scenarios.

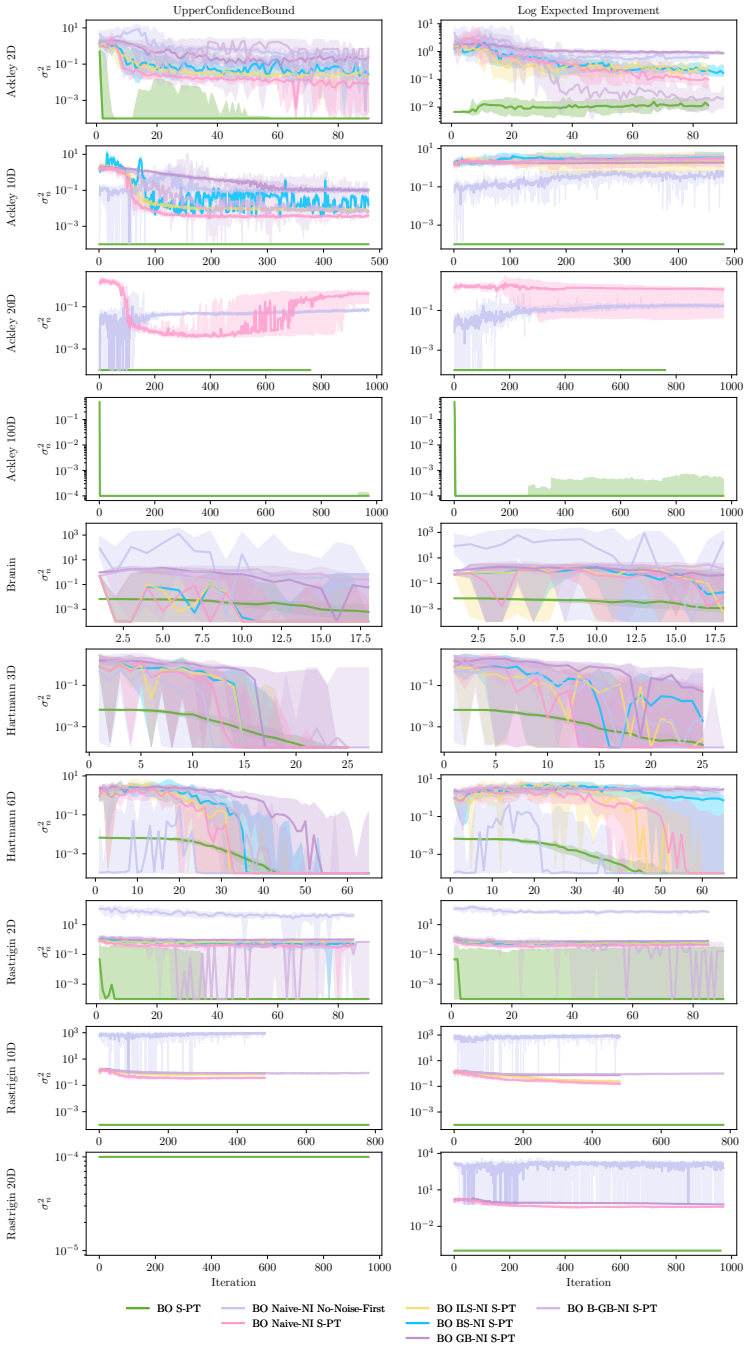

Figure A.12.: Evolution of the noise variance hyperparameter during the heuristic noise injection benchmarking across all tested scenarios.

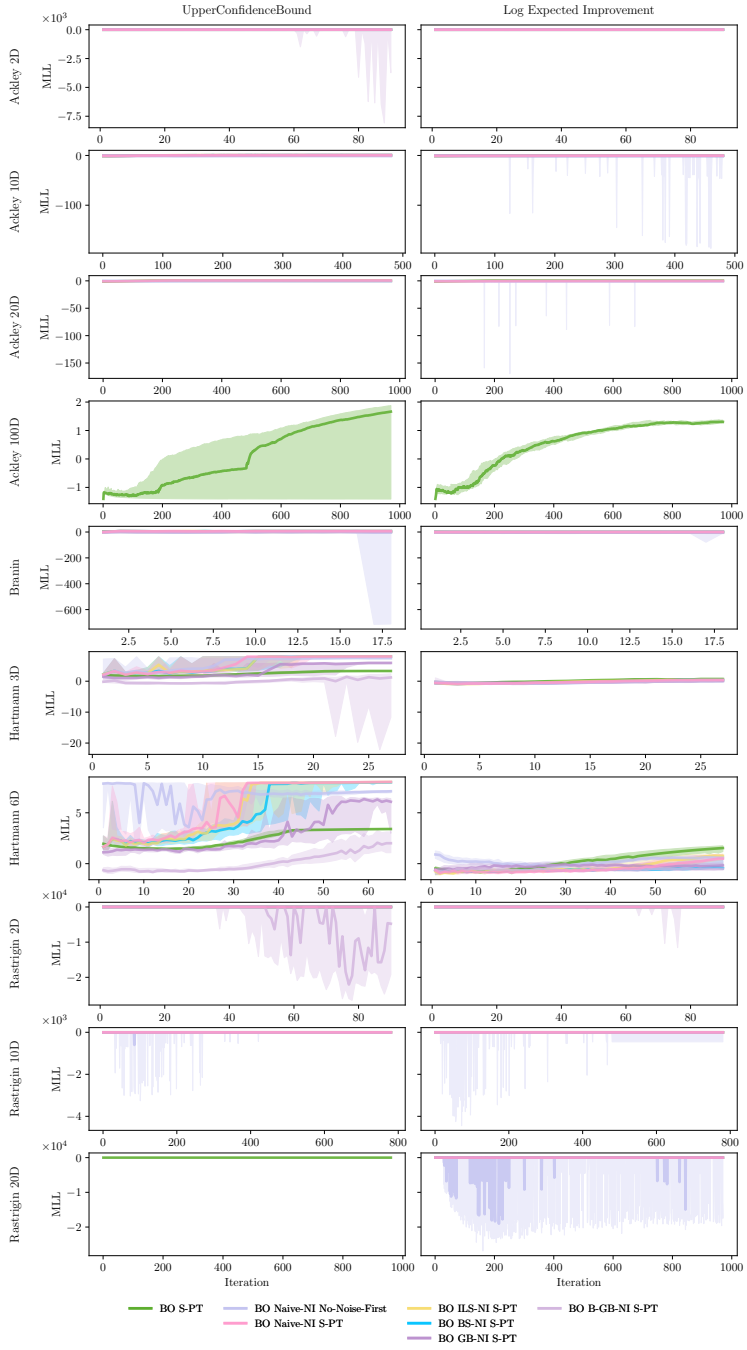

Figure A.13.: Evolution of the MLL during the heuristic noise injection benchmarking across all tested scenarios.

Trust Region Noise Injection Benchmarking

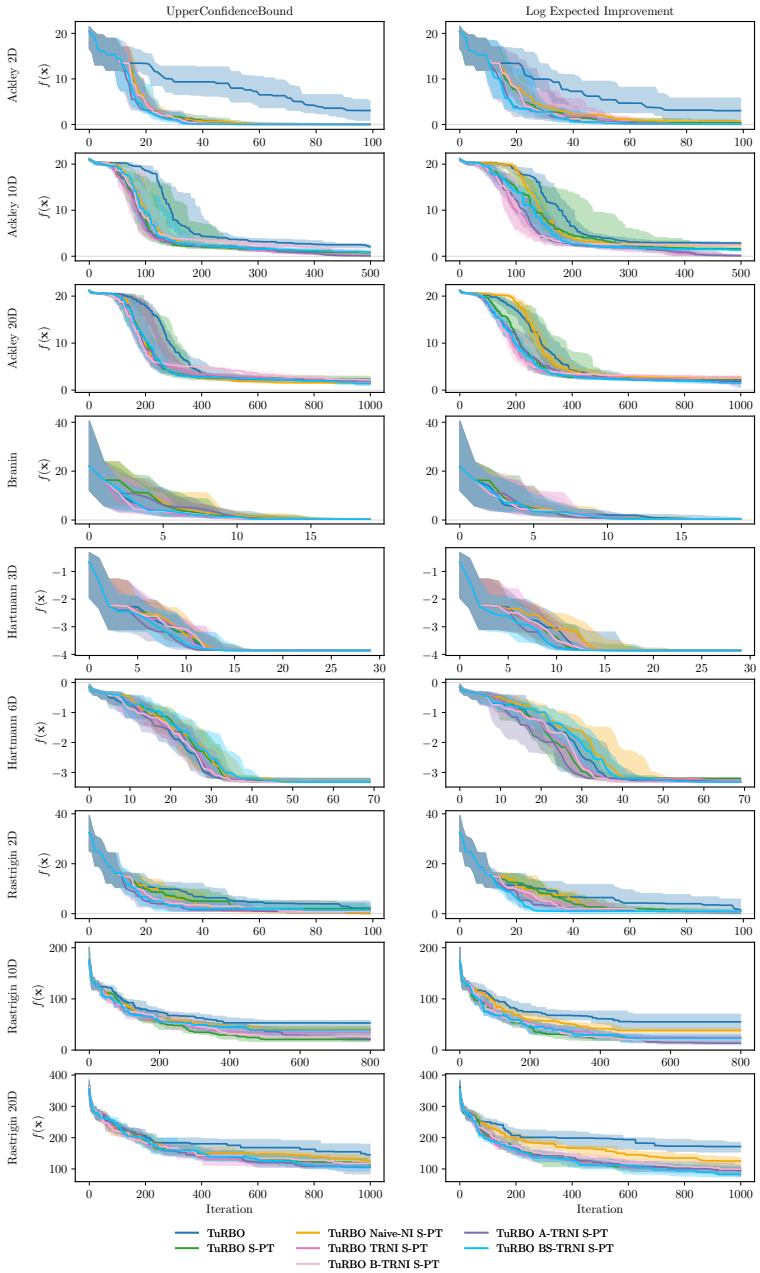

Figure A.14.: Regret trajectories for the trust region noise injection benchmarking across all tested objective functions and dimensionalities.

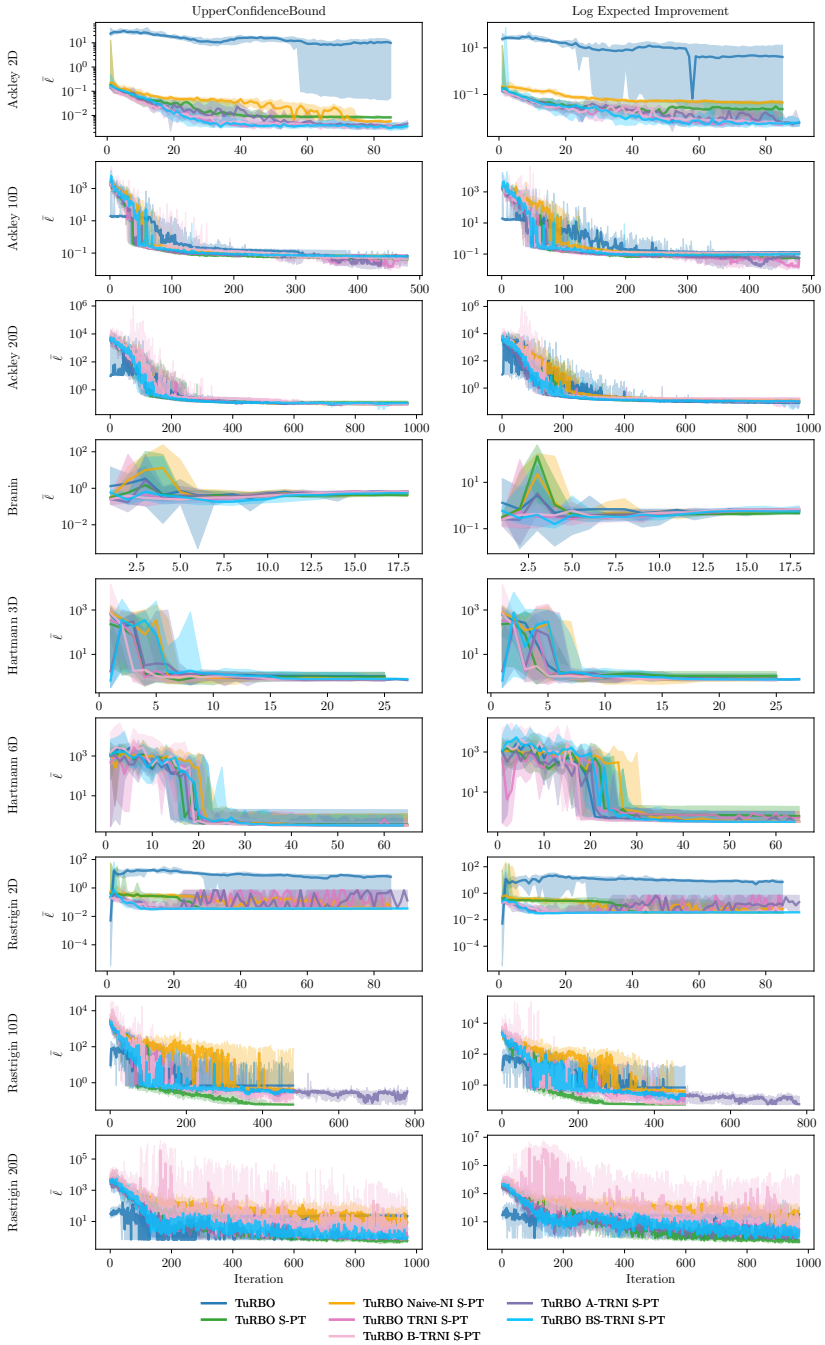

Figure A.15.: Evolution of the mean length-scale hyperparameter during the trust region noise injection benchmarking across all tested scenarios.

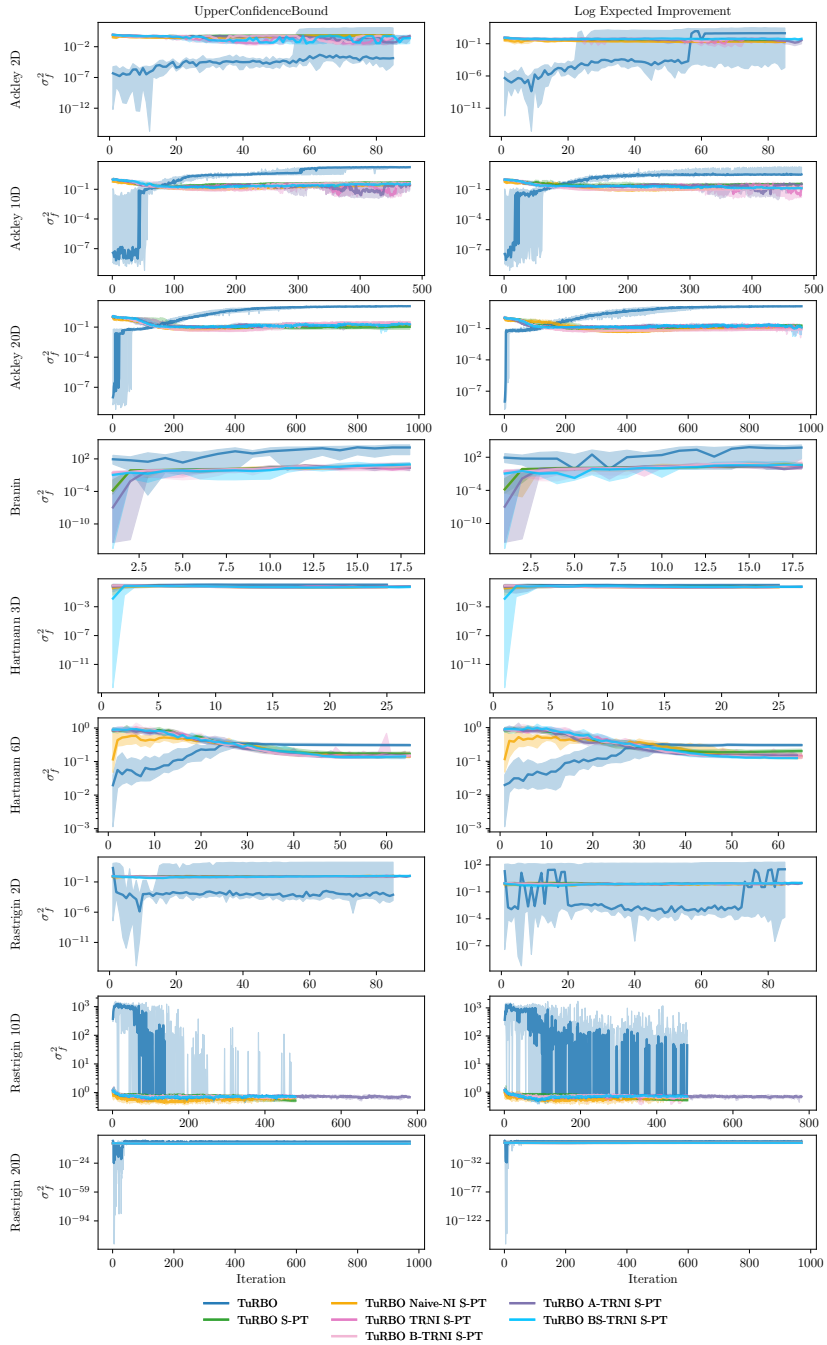

Figure A.16.: Evolution of the signal variance hyperparameter during the trust region noise injection benchmarking across all tested scenarios.

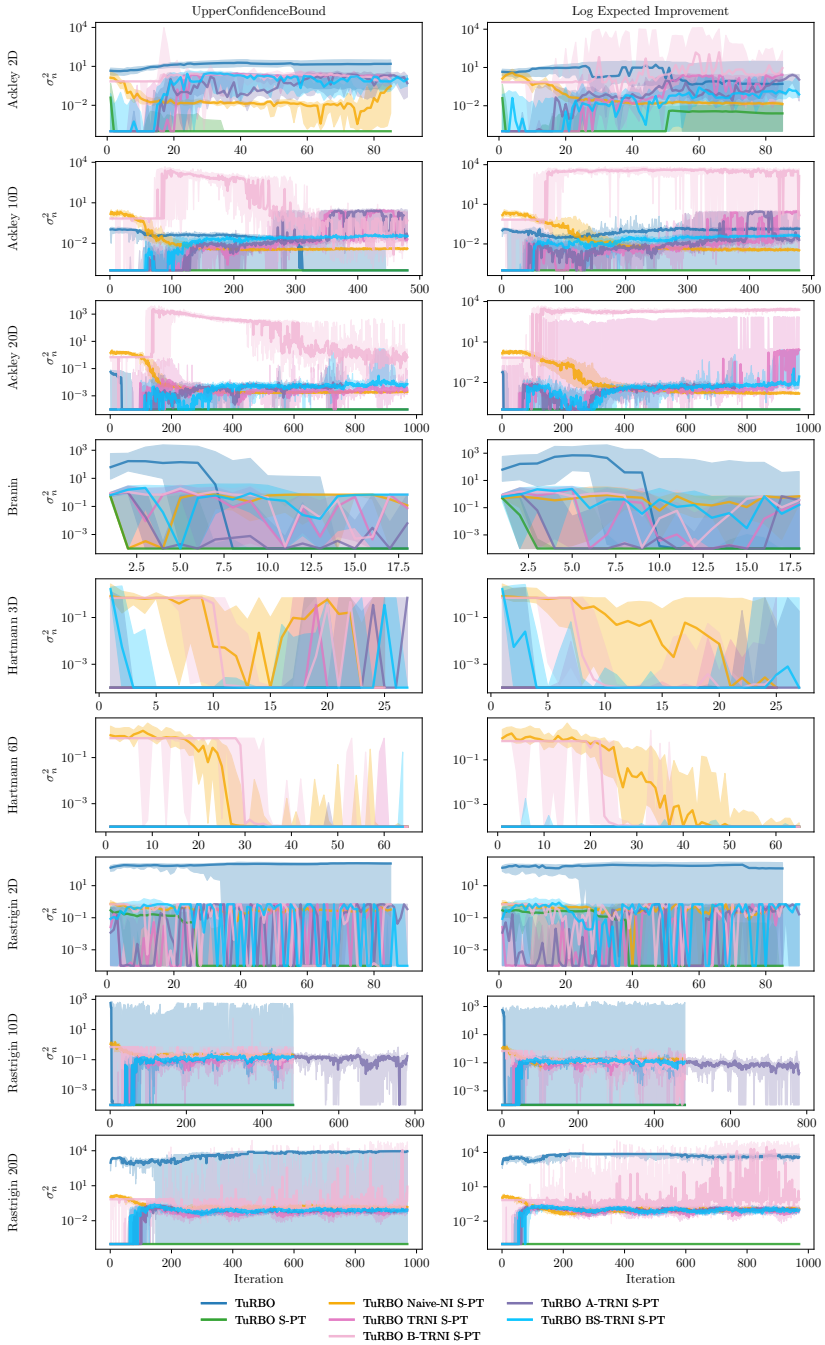

Figure A.17.: Evolution of the noise variance hyperparameter during the trust region noise injection benchmarking across all tested scenarios.

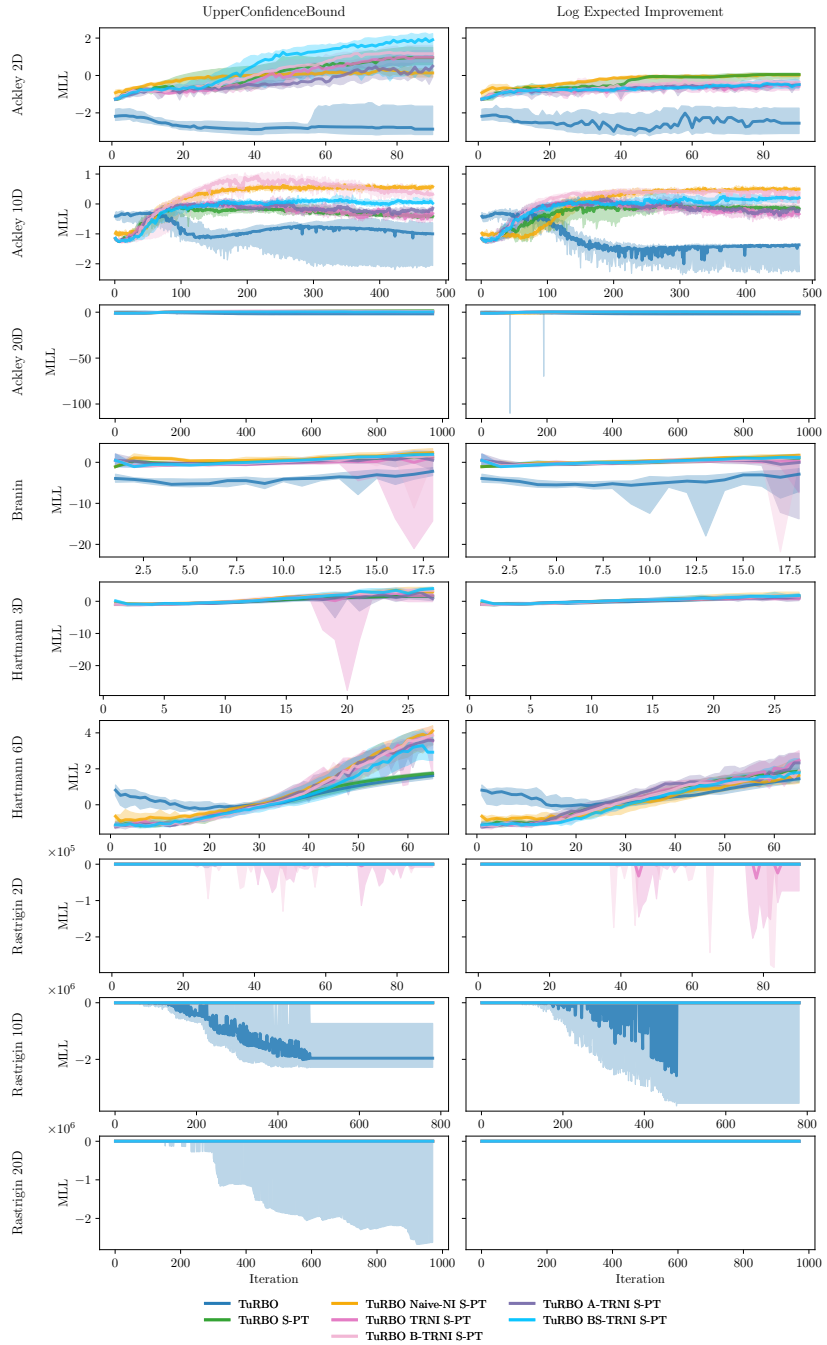

Figure A.18.: Evolution of the MLL during the trust region noise injection benchmarking across all tested scenarios.

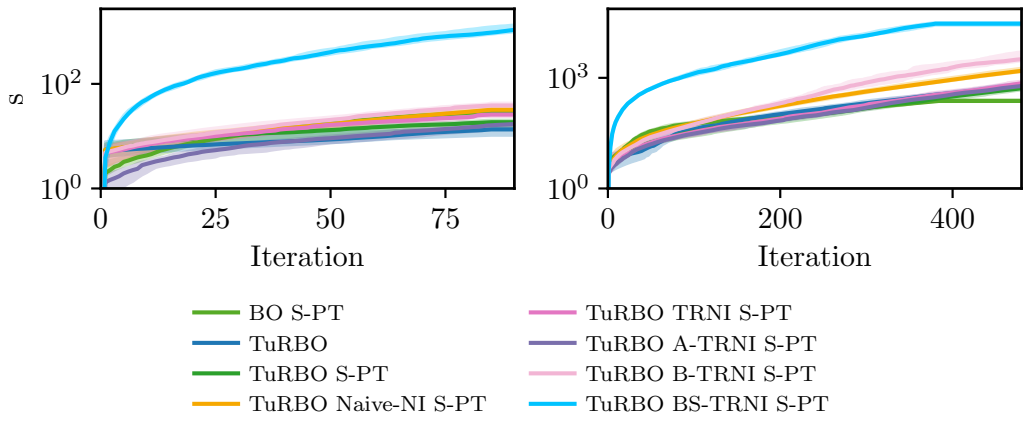

Figure A.19.: Cumulative computational runtime (log-scaled) for different TR-NI variants on the Ackley-2D (left) and Ackley-10D (right) benchmarks with UCB. The lines and shaded areas represent the median runtime and interquartile range, respectively, over 20 independent runs.

B. Hyperparameter

The following hyperparameters were used in all experiments.

Table B.1.: General hyperparameters (GP, BO, and preprocessing).

Hyperparameter	**Value**
Additive logarithmic transformation constant c	1
GP	
Initial length-scale ℓ	$\sqrt{d}$
Maximal GP fit retries	10
Minimal noise jitter $\epsilon_{\min}$	10^{-6}
BO	
UCB exploration parameter β	2
Acquisition-optimizer restarts	10
Raw samples per acquisition-optimizer restart	512
Batch size	1
Seed set	$\{0, \ldots, 19\}$ (20 runs)
Objective function	
Number of initial points (Ackley 2D, 10D, 20D, 100D)	10, 20, 40, 200
Number of initial points (Branin)	2
Number of initial points (Hartmann 3D, 6D)	3, 6
Number of initial points (Rastrigin 2D, 10D, 20D)	10, 20, 40
Noise standard deviation	0

Table B.2.: Hyperparameters for heuristic noise injection methods.

Hyperparameter	**Value**
Maximal GP fit retries for MLL calculation	1
GB-NI and B-GB-NI	
Base learning rate	0.01
Effective learning rate	base/max(1, $d/5$)
Number of iterations	300
Scheduler factor	0.8
Patience	10
ILS-NI	
Number of iterations	80
Flip fraction	0.20
Stagnation limit	10
BS-NI	
Number of iterations	Number of observed points
Beam width k	5
Discard fraction	$\frac{1}{3}$
Convergence threshold	10^{-6}
Stagnation limit	5

Table B.3.: Hyperparameters for trust region noise injection methods.

Hyperparameter	**Value**
Trust-region length τ	0.8
Minimal trust-region length $\tau_{\min}$	0.5^7
Maximal trust-region length $\tau_{\max}$	1.6
Success tolerance T_{succ}	3
Failure tolerance T_{fail}	$\left\lceil \max\left(\frac{4}{\text{batch size}}, \frac{d}{\text{batch size}}\right)\right\rceil$
Restart perturbation scale	0.1
A-TR-NI noise step $\Delta\lambda$	1
BS-TR-NI beam width k	5
BS-TR-NI stagnation limit	3

Acronyms

A-TR-NI	Additive trust-region-guided noise injection
BS-TR-NI	Beam search trust-region-guided noise injection
B-TR-NI	Binary trust-region-guided noise injection
B-GB-NI	Binary gradient-based noise injection
BO	Bayesian optimization
BS	Beam search
BS-NI	Beam search noise injection
CDF	Cumulative distribution function
EI	Expected improvement
GB-NI	Gradient-based noise injection
GP	Gaussian process
iid	Independent and identically distributed
ILS	Iterated local search
ILS-NI	Iterated local search noise injection
MLE	Maximum likelihood estimation
MLL	Marginal log likelihood
Naive-NI	Naive noise injection
OT	Outcome transformation

PDF	Probability density function
PT	Preprocessing transformation
RBF	Radial basis function
S-PT	Standardization (as a preprocessing transformation)
TR-NI	Trust-region-guided noise injection
TuRBO	Trust region Bayesian optimization
UCB	Upper confidence bound

List of Mathematical Symbols

$\alpha(\mathbf{x}; \mathcal{D})$	Acquisition function evaluating $\mathbf{x}$ given data $\mathcal{D}$
$\alpha_{\mathbf{UCB}}$	Upper confidence bound
$\alpha_{\mathbf{EI}}$	Expected improvement
β	Exploration parameter in UCB acquisition function
$\mathrm{Cov}(\cdot, \cdot)$	Covariance between two random variables
$\mathcal{D}$	Set of observed data, $\mathcal{D} = (\mathbf{x}, \mathbf{y})$
$\mathcal{D}_n$	Set of observed data after n observations
ε	Measurement error associated with an observation
$\mathbb{E}[\cdot]$	Expected value of a random variable
f	Objective function; $f \colon \mathcal{X} \to \mathbb{R}$
$f^\star$	Globally minimal value of the objective function: $f^\star = \min f(x^\star)$
$\mathcal{GP}(\cdot, \cdot)$	Gaussian process
K	Covariance matrix with entries $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
$k(\boldsymbol{x}, \boldsymbol{x}')$	Covariance or kernel function: $k(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}'))$
$\boldsymbol{\lambda}$	Noise vector for multiplicative Gaussian likelihood

ℓ	Length-scale parameter
τ	Trust region side length
η	Logit
$m(\boldsymbol{x})$	Prior mean function, $m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$
$\mu(\boldsymbol{x})$	Posterior mean function given data $\mathcal{D}$: $\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x}) \mid \mathcal{D}]$
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
$\mathcal{P}$	Probability distribution
$\mathbb{R}$	Set of real numbers
σ_f^2	Signal variance
σ_n^2	Noise variance
$\sigma^2(x)$	Posterior variance function: $\sigma^2(x) = \mathrm{Var}(f(x) \mid \mathcal{D})$
$\mathcal{TR}$	Trust region
$\mathrm{Var}(\cdot)$	Variance of a random variable
x	Putative input location
$\boldsymbol{x}$	Vector-valued input location
$\boldsymbol{x}_c$	Trust region center
$\mathcal{X}$	Domain of the objective function
y	Value resulting from an observation at $\boldsymbol{x}$

List of Figures

List of Tables

Bibliography

[1] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing* (The Springer International Series in Engineering and Computer Science). Boston, MA: Kluwer Academic Publishers, 1987, vol. 28, Definition of the Ackley test function appears on pp. 13–14.

[2] M. Altamirano, F.-X. Briol, and J. Knoblauch, "Robust and conjugate gaussian process regression," *arXiv preprint arXiv:2311.00463*, 2023.

[3] S. Ament, E. Santorella, D. Eriksson, B. Letham, M. Balandat, and E. Bakshy, "Robust gaussian processes via relevance pursuit," *Advances in Neural Information Processing Systems*, vol. 37, pp. 61 700–61 734, 2024.

[4] M. Balandat, B. Karrer, D. R. Jiang, *et al.*, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Advances in Neural Information Processing Systems 33*, 2020. [Online]. Available: `http://arxiv.org/abs/1910.06403`.

[5] G. E. Box and D. R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 26, no. 2, pp. 211–243, 1964.

[6] F. H. Branin, "Widely convergent method for finding multiple solutions of simultaneous nonlinear equations," *IBM Journal of Research and Development*, vol. 16, no. 5, pp. 504–522, 1972.

[7] P. Brunzema, A. Von Rohr, and S. Trimpe, "On controller tuning with time-varying bayesian optimization," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, IEEE, 2022, pp. 4046–4052.

[8] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," *Advances in neural information processing systems*, vol. 32, 2019.

[9] A. I. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the royal society a: mathematical, physical and engineering sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007.

[10] N. Frohner, J. Gmys, N. Melab, G. Raidl, and E.-G. Talbi, "Parallel beam search for combinatorial optimization," in *Workshop Proceedings of the 51st International Conference on Parallel Processing*, 2022, pp. 1–8.

[11] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," in *Advances in Neural Information Processing Systems*, 2018.

[12] R. Garnett, *Bayesian optimization*. Cambridge University Press, 2023.

[13] D. Ginsbourger, R. Le Riche, and L. Carraro, "Kriging is well-suited to parallelize optimization," in *Computational intelligence in expensive optimization problems*, Springer, 2010, pp. 131–162.

[14] P. Goldberg, C. Williams, and C. Bishop, "Regression with input-dependent noise: A gaussian process treatment," *Advances in neural information processing systems*, vol. 10, 1997.

[15] J. K. Hartman, "Some experiments in global optimization," *Naval Research Logistics Quarterly*, vol. 20, no. 3, pp. 569–576, 1973.

[16] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy, "An experimental investigation of model-based parameter optimisation: Spo and beyond," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 271–278.

[17] C. Hvarfner, E. O. Hellsten, and L. Nardi, "Vanilla bayesian optimization performs great in high dimensions," *arXiv preprint arXiv:2402.02229*, 2024.

[18] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

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

[20] N. Jaquier, V. Borovitskiy, A. Smolensky, A. Terenin, T. Asfour, and L. Rozo, "Geometry-aware bayesian optimization in robotics using riemannian matérn kernels," in *Conference on Robot Learning*, PMLR, 2022, pp. 794–805.

[21] P. Jylänki, J. Vanhatalo, and A. Vehtari, "Robust gaussian process regression with a student-t likelihood.," *Journal of Machine Learning Research*, vol. 12, no. 11, 2011.

[22] E. Kamrah, S. F. Ghoreishi, Z. " Ding, J. Chan, and M. Fuge, "How diverse initial samples help and hurt bayesian optimizers," *Journal of Mechanical Design*, vol. 145, no. 11, p. 111 703, 2023.

[23] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964. DOI: 10.1115/1.3653121.

[24] M. Lázaro-Gredilla and M. K. Titsias, "Variational heteroscedastic gaussian process regression.," in *ICML*, 2011, pp. 841–848.

[25] Q. V. Le, A. J. Smola, and S. Canu, "Heteroscedastic gaussian process regression," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 489–496.

[26] A. Li, A. Boyd, P. Smyth, and S. Mandt, "Detecting and adapting to irregular distribution shifts in bayesian online learning," *Advances in neural information processing systems*, vol. 34, pp. 6816–6828, 2021.

[27] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through L_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.

[28] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of metaheuristics*, Springer, 2003, pp. 320–353.

[29] B. T. Lowerre, *The harpy speech recognition system.* Carnegie Mellon University, 1976.

[30] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[31] J. Mockus, "The application of bayesian methods," in *Bayesian Approach to Global Optimization: Theory and Applications*, Springer, 1989, pp. 157–196.

[32] L. A. Rastrigin, "Systems of extremal control," *Nauka*, 1974.

[33] M. L. Santoni, E. Raponi, R. D. Leone, and C. Doerr, "Comparison of high-dimensional bayesian optimization algorithms on bbob," *ACM Transactions on Evolutionary Learning*, vol. 4, no. 3, pp. 1–33, 2024.

[34] A. Sarmadi, P. Krishnamurthy, and F. Khorrami, "High-dimensional controller tuning through latent representations," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 10 853–10 859.

[35] E. Snelson, Z. Ghahramani, and C. Rasmussen, "Warped gaussian processes," *Advances in neural information processing systems*, vol. 16, 2003.

[36] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[37] J. Snoek, K. Swersky, R. Zemel, and R. Adams, "Input warping for bayesian optimization of non-stationary functions," in *International conference on machine learning*, PMLR, 2014, pp. 1674–1682.

[38] P. Sollich, "Gaussian process regression with mismatched models," *Advances in Neural Information Processing Systems*, vol. 14, 2001.

[39] F. Triggiano and M. Romito, "Gaussian processes based data augmentation and expected signature for time series classification," *IEEE Access*, 2024.

[40] A. Vehtari, A. Gelman, and J. Gabry, "Practical bayesian model evaluation using leave-one-out cross-validation and waic," *Statistics and computing*, vol. 27, no. 5, pp. 1413–1432, 2017.

[41] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, *et al.*, "Diverse beam search: Decoding diverse solutions from neural sequence models," *arXiv preprint arXiv: 1610.02424*, 2016.

[42] Z. Wang, G. E. Dahl, K. Swersky, *et al.*, "Pre-trained gaussian processes for bayesian optimization," *Journal of Machine Learning Research*, vol. 25, no. 212, pp. 1–83, 2024.

[43] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[44] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial intelligence and statistics*, PMLR, 2016, pp. 370–378.

[45] G. Wynne, F.-X. Briol, and M. Girolami, "Convergence guarantees for gaussian process means with misspecified likelihoods and smoothness," *Journal of Machine Learning Research*, vol. 22, no. 123, pp. 1–40, 2021.

[46] Z. Xu, H. Wang, J. M. Phillips, and S. Zhe, "Standard gaussian process is all you need for high-dimensional bayesian optimization," in *The Thirteenth International Conference on Learning Representations*, 2024.

[47] O. Yadan, *Hydra - a framework for elegantly configuring complex applications*, Github, 2019. [Online]. Available: `https://github.com/facebookresearch/hydra`.