Universiteit Leiden

# Reducing Stagnation in PCA-Assisted Bayesian Optimization

Benas Klioštoraitis (s3837424)

Supervisors:
Elena Raponi & Iván Olarte Rodríguez

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 1, 2026

**Abstract**

Bayesian Optimization is used for optimizing expensive black-box functions, but it can become difficult to apply when the search space is high-dimensional. PCA-assisted Bayesian Optimization addresses this by building a lower-dimensional latent space from previously evaluated points and performing the optimization in this reduced space. However, the PCA subspace depends on the evaluations collected so far. If some important directions are not represented by these points, the optimizer may keep sampling through an incomplete latent space and stagnate.

This thesis studies whether PCA-BO can be made more robust by adding controlled exploration outside the learned PCA subspace. The main method is Dynamic Orthogonal PCA-BO, which follows the usual PCA-BO step by default, but activates an orthogonal perturbation when progress-based stagnation signals indicate that the search is no longer improving. A trust-region extension is also studied, where the same dynamic orthogonal mechanism is applied locally inside an adaptive TuRBO-style trust region.

The methods are evaluated on BBOB benchmark functions in 10, 20, and 40 dimensions, using multiple instances and random seeds. Algorithms are compared against standard Bayesian Optimization, vanilla PCA-BO, and TuRBO-1 under matched evaluation budgets. The experiments are designed to show whether orthogonal exploration can reduce PCA-BO stagnation, and whether adding a trust-region mechanism improves the method.

The results show that Dynamic Orthogonal PCA-BO can reduce stagnation in some cases, but its benefit strongly depends on the benchmark function. The clearest improvements appear on functions with weaker global structure, where plain PCA-BO is more likely to stagnate. The trust-region extension improves the dynamic method on several functions, but can also restrict the broader exploration that makes the global orthogonal method useful. Overall, controlled orthogonal exploration is a useful extension of PCA-BO, but it does not fully match the performance of strong local methods such as TuRBO-1.

# Contents

# 1 Introduction

Bayesian Optimization (BO) is an algorithm for optimizing expensive black-box functions. In black-box optimization, the function is not known, but it can be evaluated. Such problems appear in many real-world settings, including hyperparameter tuning, simulation-based optimization, engineering design, robotics, and physical experiments. In these cases, each evaluation may require significant time and/or computational resources, making it essential to find good solutions with a limited number of evaluations.

BO addresses this problem by using the data collected from previous evaluations to decide where to sample next. BO builds a probabilistic surrogate model of the objective function. This model is often a Gaussian Process, which provides both a prediction of the objective value and an estimate of uncertainty for it. An acquisition function is then used to select the next evaluation point by balancing exploitation of good regions with exploration of uncertain regions.

Despite performing well in low-dimensional problems, BO becomes more difficult to use in high-dimensional spaces. As the number of dimensions increases, the search space grows rapidly and each evaluation provides less coverage of the full domain. This is commonly referred to as the curse of dimensionality. The surrogate model also becomes harder to fit accurately, and optimizing the acquisition function becomes more challenging.

These challenges were addressed by Raponi et al. [8], who proposed PCA-assisted Bayesian Optimization (PCA-BO). Their method is based on the assumption that, although the original problem is high-dimensional, the objective function may vary mainly along a smaller number of important directions. PCA-BO uses Principal Component Analysis (PCA) on already evaluated points to estimate such directions. The optimization is then performed in the resulting lower-dimensional latent space, which can reduce computational cost while maintaining convergence.

However, the same projection that makes PCA-BO efficient can also make it biased. Since the PCA subspace is learned from previously evaluated points, it may miss directions that are useful later in the search. The optimizer can then keep proposing candidates through an incomplete latent space, which may lead to stagnation.

This thesis studies whether this can be reduced by adding controlled exploration outside the learned PCA subspace. The main method, Dynamic Orthogonal PCA-BO, keeps the usual PCA-BO step as the default, but activates orthogonal perturbations when convergence tracking signals indicate stagnation.

The idea of adapting exploration during optimization is related to work such as Self-Adjusting Weighted Expected Improvement (SAWEI) [2]. SAWEI adjusts the exploration-exploitation behavior of the acquisition function during the optimization process. This thesis follows a similar motivation in the context of PCA-BO by studying how exploration should change if the optimizer appears to stagnate. Following this idea, this thesis studies a dynamic orthogonal exploration strategy, where orthogonal perturbations are activated when the optimization appears to stagnate.

A second direction considered in this thesis is local trust-region search. In high-dimensional optimization, it can be useful to focus the search around promising regions, while still keeping the region size adaptable. This is also relevant for multimodal landscapes, where different promising regions may exist in different parts of the search space. The TuRBO method, proposed by Eriksson et al. [3], addresses this by maintaining adaptive trust regions around promising points.

The goal of this thesis is to investigate whether PCA-assisted Bayesian Optimization can be made more robust by combining controlled exploration outside the learned PCA subspace

with TuRBO-style local search. The focus is on reducing stagnation of PCA-BO and improving performance on high-dimensional benchmark optimization problems.

*How can PCA-assisted Bayesian Optimization be extended to reduce stagnation in high-dimensional optimization?*

### 1.1 Thesis Overview

This bachelor thesis was carried out at the Leiden Institute of Advanced Computer Science (LIACS) under the supervision of Elena Raponi and Ivan Olarte Rodriguez.

The thesis is structured as follows. Section 2 introduces the relevant background on Bayesian Optimization, PCA-assisted Bayesian Optimization, adaptive exploration, and trust-region Bayesian Optimization. Section 3 presents the main contribution of the thesis, Dynamic Orthogonal PCA-BO, including the orthogonal exploration mechanism, stagnation detection rule, experimental setup, and results. Section 4 describes the trust region extension, where the same dynamic orthogonal idea is applied locally inside an adaptive trust region. Section 5 discusses the results and limitations, and Section 6 concludes the thesis.

## 2 Related Work

### 2.1 Bayesian Optimization

Bayesian Optimization (BO) is an optimization method for expensive black-box objective functions. It is especially useful when function evaluations are costly and the optimization must be performed under a limited evaluation budget [4]. BO uses the previously evaluated points to build a probabilistic model of the objective function. This model is then used to decide which point should be evaluated next.

Let the objective function be denoted by

$$f : \mathcal{X} \subset \mathbb{R}^D \to \mathbb{R},$$

where $\mathcal{X}$ is the feasible search domain and $D$ is the dimensionality of the input space. The output $f(x)$ is a scalar objective value. In this thesis, the benchmark functions are treated as minimization problems. The goal is therefore to find a point $x^* \in \mathcal{X}$ such that

$$f(x^*) \leq f(x)$$

for all feasible points $x \in \mathcal{X}$, while using as few function evaluations as possible.

The general BO loop starts by evaluating an initial set of points (Initial Design of Experiments). These observations are used to fit a surrogate model, in this thesis we use a Gaussian Process. An acquisition function then, based on the outcome from the surrogate model, selects the next point to evaluate. After this point is evaluated, the new evaluation is added to the data and during next iteration the surrogate model is updated. This process loops until the evaluation budget is exhausted.

This loop follows the standard Bayesian Optimization procedure described in Frazier [4]: a surrogate model is updated after each evaluation, and the next point is selected by maximizing an

acquisition function. Algorithm 1 summarizes this general procedure for the minimization setting used in this thesis.

**Algorithm 1** Bayesian Optimization

**Require:** Objective function $f$, search domain $\mathcal{X}$, evaluation budget $N$, initial design size $n_0$, acquisition function $a$

Generate initial design $X = \{x_1, \ldots, x_{n_0}\} \subset \mathcal{X}$
Evaluate $y_i = f(x_i)$ for $i = 1, \ldots, n_0$
**for** $n = n_0, \ldots, N-1$ **do**
    Fit surrogate model $M_n$ using $(X, y)$
    Select next point $x_{n+1} \leftarrow \arg\max_{x \in \mathcal{X}} a(x; M_n)$
    Evaluate $y_{n+1} \leftarrow f(x_{n+1})$
    Update $X \leftarrow X \cup \{x_{n+1}\}$ and $y \leftarrow y \cup \{y_{n+1}\}$
**end for**
**return** Best evaluated point

The next evaluation point is selected by considering the predicted objective value and the uncertainty estimate for that point. Having both predicted objective value and the uncertainty estimate, gives a way to balance exploration and exploitation.

### 2.1.1 Design of Experiments

Before the BO loop, the algorithm needs an initial set of evaluated points. This initial sampling phase is often called the Design of Experiments (DoE). The purpose of the DoE is to provide the surrogate model with enough information to construct the first model of the objective function.

Poor initial design can strongly influence the behaviour of the algorithm. If, for example, the initial points are concentrated in only a part of the domain, the surrogate model may have high uncertainty elsewhere and may also learn a misleading representation of the objective.

Purely random initial sampling may provide unbalanced coverage of the search domain, particularly in higher-dimensional settings. For this reason, space-filling designs are used. Two widely used methods for DoE are Latin Hypercube Sampling and Sobol sequences [7, 9].

Latin Hypercube Sampling splits the range of each dimension into intervals and spreads the samples across them. This gives better coverage for each direction than purely random sampling. Sobol sequences are deterministic low-discrepancy sequences designed to distribute points evenly over the unit cube. This makes the samples more evenly distributed than ordinary random sampling.

In this thesis, initial design is generated using Latin Hypercube Sampling in the original space, before any PCA projection is applied. This follows the PCA-BO setting, where the first PCA subspace is learned from the initially evaluated points. We also use Sobol sampling for the TuRBO-style part of the method, where it is used to generate restart points.

### 2.1.2 Gaussian Process Surrogate Models

A surrogate model is used in BO to approximate the unknown objective function using the observations collected so far. A common surrogate model is a Gaussian Process (GP), which defines

a probability distribution over possible functions [11]. The advantage of a GP is that it provides both a prediction and an uncertainty estimate.

A Gaussian Process is written as

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

where $m(x)$ is the mean function and $k(x, x')$ is the kernel function. The mean function describes the prior expected value of the function, while the kernel describes how function values at two input points are related. In many BO settings, a zero or constant mean function is used, and the structure of the function is mainly represented through the kernel.

Given evaluated input points

$$X = \{x_1, \ldots, x_n\}$$

and corresponding objective values

$$y = \{f(x_1), \ldots, f(x_n)\},$$

the GP gives a posterior predictive distribution at a new point $x$. This distribution is commonly described by a posterior mean $\mu(x)$ and posterior variance $\sigma^2(x)$. The posterior mean represents the model's current best estimate of the objective value, while the posterior variance represents the uncertainty of that estimate.

The kernel function is important because it determines how information from evaluated points is used to describe unevaluated points. For example, if two input points are close according to the kernel, their objective values are expected to be related. Most used kernels in BO include the squared exponential kernel and Matérn kernels. The Matérn kernel is often used because it can model functions that are less smooth than those assumed by the squared exponential kernel.

In this thesis, Gaussian Processes are used as surrogate models in all optimization methods considered.

### 2.1.3 Acquisition Functions

The acquisition function is used to choose the next point for evaluation. It combines the predicted mean and uncertainty to balance exploitation and exploration. For minimization problems, exploitation benefits from points with low predicted objective values, while exploration favours points where the model has higher uncertainty.

One oftenly used acquisition function is Expected Improvement (EI) [6]. EI measures the expected improvement over the best objective value observed so far. Let $f_{\min}$ denote the current best observed value. For minimization, the improvement at a candidate point $x$ can be written as

$$I(x) = \max(f_{\min} - f(x), 0).$$

Since $f(x)$ is unknown before evaluation, EI takes the expectation of this improvement using the GP posterior distribution. If the posterior mean is low, the point may improve the current best value. If the posterior uncertainty is high, the point may also have a chance of improvement. Therefore, EI naturally combines exploitation and exploration.

For a Gaussian predictive distribution with mean $\mu(x)$ and standard deviation $\sigma(x)$, EI for minimization can be written as

$$\mathrm{EI}(x) = (f_{\min} - \mu(x))\Phi(z) + \sigma(x)\phi(z),$$

where

$$z = \frac{f_{\min} - \mu(x)}{\sigma(x)}.$$

Here, $\Phi$ is the cumulative distribution function of the standard normal distribution and $\phi$ is its probability density function.

Another common type of acquisition functions is based on confidence bounds [10]. For minimization, a lower confidence bound can be written as $\mathrm{LCB}(x) = \mu(x) - \kappa\sigma(x)$, where $\kappa$ controls the strength of exploration. A larger value of $\kappa$ gives more importance to uncertainty and therefore encourages exploration.

During acquisition optimization, the acquisition function is evaluated on candidate points proposed during the acquisition optimization step. Each candidate receives an acquisition score based on the GP mean and uncertainty, and the best-scoring candidate is selected for evaluation on the real objective function.

### 2.1.4 Limitations of Bayesian Optimization in High Dimensions

Although BO performs well on many low-dimensional black-box optimization problems, it becomes harder to apply as the dimensionality increases. This is commonly known as the curse of dimensionality. As the number of dimensions grows, the volume of the search space increases rapidly, so a fixed evaluation budget covers only a very small part of the domain.

This affects both the surrogate model and the acquisition step. The surrogate model becomes harder to fit accurately because the evaluated points are sparse relative to the full search space. For GPs, this means that much of the domain is not observed, leading to high uncertainty and less reliable predictions. At the same time, optimizing the acquisition function becomes more difficult because it must be searched over more variables and a more complex landscape.

The exploration-exploitation trade-off also becomes harder to manage. In high-dimensional spaces, there are many uncertain regions, but exploring all of them is impossible under a limited evaluation budget. This is especially problematic for multimodal functions, where promising basins may be located in different parts of the domain. A global surrogate model may fail to represent these regions equally well, causing the optimizer to over-explore bad areas or exploit suboptimal regions.

These limitations motivate approaches that either reduce the effective dimensionality of the problem or make the surrogate modeling more local. The following sections discuss both directions.

## 2.2 PCA-Assisted Bayesian Optimization

### 2.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction method that finds directions with the largest variation in a data set. These directions are called principal components. The first principal component explains the largest amount of variation in the data, the next ones explain the largest remaining variation while being orthogonal to the last one, and so on.

Let $X \in \mathbb{R}^{n \times D}$ be a data matrix containing $n$ evaluated points in a $D$-dimensional search space. PCA first centers the data by subtracting the sample mean $\mu$. The covariance matrix of the centered data is then computed as

$$C = \frac{1}{n-1} X_c^\top X_c,$$

where $X_c = X - \mu$. The principal components are obtained using the eigendecomposition of this covariance matrix. The eigenvectors give the principal directions, while the corresponding eigenvalues describe how much variation is explained by each direction.

The eigenvalues are then sorted in decreasing order. PCA can then reduce the dimensionality by keeping only the first $r$ principal components, where $r < D$. In PCA-BO, $r$ is chosen so that the selected components explain a 95% of the total variance, corresponding to $\alpha = 0.95$

$$\frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{D} \lambda_i} \geq \alpha,$$

where $\lambda_i$ are the eigenvalues sorted from largest to smallest.

If the selected principal components are collected in a matrix $P_r \in \mathbb{R}^{r \times D}$, then a point from the original space can be projected into the lower-dimensional PCA space. This reduced space is also called the latent space.

### 2.2.2 PCA-BO Algorithm

PCA-assisted Bayesian Optimization (PCA-BO), proposed by Raponi et al. [8], uses PCA to reduce the dimensionality of the search problem before applying Bayesian Optimization. The method is based on the idea that, even if the original problem is high-dimensional, the objective function may vary mainly along a smaller number of directions.

The algorithm starts with an initial Design of Experiments in the original search space. After these points have been evaluated, principal components are calculated from the initial points. Raponi et al. use a weighted PCA procedure, where points with better objective values receive larger weights. For minimization, the best point receives the highest weight. If $q_i$ denotes the rank of point $i$ according to its objective value, the rank-based pre-weight is

$$\tilde{w}_i = \log(n) - \log(q_i),$$

and the normalized weight is

$$w_i = \frac{\tilde{w}_i}{\sum_{j=1}^{n} \tilde{w}_j}.$$

These weights make the PCA directions depend more on their objective values, rather than just on the geometry of the sampled points. As a result, the learned PCA subspace is influenced more strongly by points that currently appear promising.

Algorithm 2 summarizes how PCA-BO modifies the Bayesian Optimization loop from Algorithm 1. The main difference is that the surrogate model and acquisition function are fitted and optimized in the PCA latent space, while the objective function is still evaluated in the original search space.

**Algorithm 2** PCA-Assisted Bayesian Optimization

**Require:** Objective function $f$, search domain $\mathcal{X}$, evaluation budget $N$, initial design size $n_0$, acquisition function $a$

Generate initial design $X = \{x_1, \ldots, x_{n_0}\} \subset \mathcal{X}$
Evaluate $y_i = f(x_i)$ for $i = 1, \ldots, n_0$
**for** $n = n_0, \ldots, N-1$ **do**
    Compute rank-based weights $w_i$ from objective values $y$
    Fit weighted PCA on $X$ and select rank $r$
    Project evaluated points into latent space: $Z_r \leftarrow \textsc{Project}(X, P_r)$
    Fit surrogate model $M_n$ using $(Z_r, y)$
    Select latent candidate $z_{n+1} \leftarrow \arg\max_z a(z; M_n)$
    Map candidate back to original space: $x_{n+1} \leftarrow \textsc{InverseProject}(z_{n+1}, P_r)$
    Evaluate $y_{n+1} \leftarrow f(x_{n+1})$
    Update $X \leftarrow X \cup \{x_{n+1}\}$ and $y \leftarrow y \cup \{y_{n+1}\}$
**end for**
**return** Best evaluated point

Because the acquisition function is optimized in the PCA latent space, a latent candidate may map back to a point outside the original box constraints. Raponi et al. [8] handle this using penalized Expected Improvement: feasible mapped points are scored using Expected Improvement, while infeasible mapped points receive a penalty. This encourages the optimizer to select candidates that are promising in the latent space and valid in the original domain.

### 2.2.3 Stagnation and Intrinsic Biases of PCA-BO

The main advantage of PCA-BO is that it can increase the efficiency of Bayesian Optimization in high-dimensional problems. The Gaussian Process is fitted in a lower-dimensional latent space, and the acquisition function is optimized over fewer variables. This can improve scalability when the learned PCA subspace captures the important directions of the objective function.

However, PCA-BO also introduces a possible bias. The PCA subspace is learned from the points that have already been evaluated. If these points do not represent all important directions of the objective function, the learned subspace may be incomplete. Since future candidates are then generated using the same subspace, the optimizer may continue sampling in directions that reinforce the current PCA model.

This can lead to stagnation and lack of exploration in the search space. In this context, stagnation means that the optimizer keeps evaluating new points, but the best objective value stops meaningfully improving. The method may still be active, but the points it is proposing are restricted by the latent space that does not have the directions needed to escape the current region.

## 2.3 Adaptive Exploration in Bayesian Optimization

The performance of Bayesian Optimization strongly depends on the balance between exploration and exploitation. Exploitation means sampling points that the surrogate model is certain to have good objective values, while exploration means sampling points in uncertain or less explored parts

of the search space. This balance is controlled by the acquisition function. Depending on the acquisition function, the optimizer may behave more greedily by focusing on expected improvement, or more exploratively by giving more importance to uncertainty.

A fixed exploration-exploitation balance is not always ideal. At the start of the optimization, the surrogate model is based on only a few observations, so more exploration may be useful. Later in the run, when promising regions have been identified, stronger exploitation may be preferred. However, if the optimizer stops improving, increasing exploration again may help it escape from a poor local region. This motivates adaptive exploration strategies, where the behaviour of the optimizer changes during the run based on the observed progress.

One example of this idea is Self-Adjusting Weighted Expected Improvement (SAWEI), proposed by Benjamins et al. [2]. SAWEI adjusts the search behaviour during the optimization. The adjustment is based on a convergence criterion related to Upper Bound Regret (UBR). When this regret estimate appears to have converged, the method changes the exploration-exploitation behaviour depending on the recent search progress.

This idea is relevant for PCA-BO because stagnation can also be interpreted as a failure of exploration. When PCA-BO repeatedly samples through the currently learned latent space, it may continue exploiting directions that are already represented by the PCA projection. Adaptive exploration suggests that the optimizer should react when progress slows down. However, in PCA-BO, simply changing the acquisition function may not be enough, because the search is still restricted by the learned PCA subspace.

For this reason, this thesis uses SAWEI as motivation for adaptive exploration in PCA-BO. The proposed method uses progress-based signals, including UBR plateauing and stagnation of the best observed value, to decide when to introduce orthogonal perturbations outside the current PCA subspace.

## 2.4 Trust-Region Bayesian Optimization

Another way to improve BO in high-dimensional problems is to make the search local. Instead of relying on one global surrogate model over the whole domain, Trust-Region Bayesian Optimization (TuRBO), proposed by Eriksson et al. [3], maintains local trust regions around promising points. The motivation is that a Gaussian Process can often model the objective more accurately in a smaller region than over the full high-dimensional search space.

In TuRBO, candidate points are generated inside a trust region. The size of this region is adapted during the optimization: it expands when the search is making progress and shrinks when repeated evaluations fail to improve the best value. If the region becomes too small, it is restarted elsewhere in the search space. This gives the optimizer a way to focus locally while still recovering from unproductive regions.

The original TuRBO framework can use either one trust region or multiple trust regions. In the multi-region setting, different regions can explore different parts of the search space, while poorly performing regions can shrink and restart. Candidate selection in TuRBO is commonly performed using Thompson sampling from a local Gaussian Process posterior.

This local modelling idea is relevant to this thesis because PCA-BO can suffer from a global latent-space bias. If one PCA projection is learned from all evaluated points, it may not represent the useful local structure of the objective equally well across the domain. A trust-region mechanism provides a way to apply PCA-BO locally: the trust region controls where the search happens, while

PCA-BO controls how the candidate is generated within that region.

## 2.5 Localized PCA-BO and Previous Thesis Work

Previous thesis work by Gregánová [5] studied how Raponi et al.'s PCA-BO can be advanced by applying the algorithm locally rather than in the whole space. The proposed method, called Localized PCA-assisted Bayesian Optimization (LPCA-BO), combines PCA-BO with a TuRBO-inspired trust-region logic. The trust region searches the part of the space that is currently considered best (or unexplored), and PCA is fitted in the TuRBO-like space using points from this local region.

In LPCA-BO, a trust region is placed around the current best solution. Points inside this region are selected and used to construct a local PCA projection. A Gaussian Process is then trained in the resulting lower-dimensional local PCA space, and the acquisition function is optimized in this reduced space. After a candidate is selected, it is mapped back to the original space and evaluated. If the trust region becomes too small, the method restarts the local search in another part of the domain.

The main motivation behind LPCA-BO is that local dimensionality reduction may describe the objective better than a global PCA projection. For example, in multimodal functions, different basins may have different important directions. A PCA subspace learned globally may average over these regions and fail to represent any of them well. By fitting PCA locally, LPCA-BO aims to capture the structure of the currently promising region more accurately.

Gregánová reports that LPCA-BO can converge faster and use less CPU time than standard PCA-BO, especially on functions where one global PCA projection is not enough to describe the search space well. This suggests that PCA-BO can benefit from using local PCA projections instead of relying only on one global projection.

The method studied in this thesis builds on the same general motivation, but focuses on a different limitation. LPCA-BO addresses the bias of global PCA by fitting PCA inside a local trust region. In this thesis, the trust-region idea is combined with dynamic orthogonal exploration. The local PCA subspace is still used as the main search space, but when stagnation is detected, the method also explores directions orthogonal to the current PCA subspace. Therefore, the proposed method does not only localize PCA-BO, but also adds a mechanism for testing directions that the learned local PCA projection may have missed.

# 3 Dynamic Orthogonal PCA-BO

This section introduces Dynamic Orthogonal PCA-BO, the main method studied in this thesis. It extends the PCA-BO loop with a conditional orthogonal exploration step, using the stagnation signals described below to decide when this step is applied.

## 3.1 Methodology

### 3.1.1 Orthogonal Exploration

Let $P_r \in \mathbb{R}^{r \times D}$ denote the matrix containing the $r$ PCA directions used by PCA-BO, where $D$ is the original dimensionality and $r < D$ is the latent dimensionality selected by PCA. The rows of

$P_r$ span the current PCA search subspace. To explore directions outside this subspace, the method constructs its orthogonal complement.

For this, the PCA basis is written as $V_r = P_r^\top$, where the columns of $V_r$ are the PCA directions. The projection onto the orthogonal complement is

$$P_\perp = I - V_r V_r^\top,$$

where $I$ is the $D$-dimensional identity matrix. Applying this projection removes the part of a vector that lies in the PCA subspace and keeps only the orthogonal component.

The method starts from a candidate point $x_{\text{PCA}}$ generated by the regular PCA-BO step. When orthogonal exploration is active, a Gaussian perturbation $\epsilon$ is sampled in the original $D$-dimensional space and projected onto the orthogonal complement,

$$\epsilon_\perp = \epsilon P_\perp.$$

The evaluated point is then

$$x_{\text{eval}} = \text{clip}\left(x_{\text{PCA}} + \epsilon_\perp\right),$$

where the clipping keeps the point inside the valid global search bounds. The perturbation scale is controlled by a dynamic parameter $\sigma_t$, which is multiplied by the average width of the search bounds so that the perturbation size is relative to the domain.

If orthogonal exploration is not active, the original PCA-BO candidate is evaluated directly. If the PCA rank is equal to the original dimensionality, no orthogonal complement exists and the method also falls back to the normal PCA-BO candidate.

### 3.1.2 Dynamic Stagnation Detection

Orthogonal exploration is not applied at every iteration. Instead, the method activates it only when the search appears to be stagnating. This follows the adaptive exploration idea discussed in Section 2.3: when the optimizer stops making progress, the search behaviour should change.

Two signals are used to detect stagnation. The first checks whether the best objective value has failed to improve over a fixed number of iterations. The second is based on the plateauing of an Upper Bound Regret style quantity computed from the surrogate model.

Let $\mu_t(z)$ and $\sigma_t(z)$ denote the posterior mean and standard deviation of the Gaussian Process in the current PCA latent space. The method defines confidence bounds as

$$\text{UCB}_t(z) = \mu_t(z) + \beta_t \sigma_t(z), \qquad \text{LCB}_t(z) = \mu_t(z) - \beta_t \sigma_t(z).$$

The UBR estimate compares the best upper confidence bound among the already evaluated latent points with the best lower confidence bound that can be found in the latent search space:

$$\text{UBR}_t = \min_{z \in Z_t} \text{UCB}_t(z) - \min_{z \in \mathcal{Z}_t} \text{LCB}_t(z),$$

where $Z_t$ denotes the evaluated points projected into the current PCA latent space, and $\mathcal{Z}_t$ denotes the current latent search region. Intuitively, a large value indicates that the model still sees possible improvement somewhere in the latent search space, while a plateauing value suggests that the current latent search has become less informative.

In the implementation, the best-value stagnation check uses a patience window of 10 iterations. If the improvement in the best observed value over this window is smaller than $10^{-8}$, this condition is considered active. The UBR signal is smoothed using a moving interquartile mean with window size 7. A plateau is detected when the latest smoothed gradient becomes small relative to the largest previous gradient. The default implementation requires both signals to be active, and this must happen for two consecutive checks before orthogonal exploration is triggered.

The strength of orthogonal exploration is controlled by a dynamic perturbation scale $\sigma_t$. It starts at 0.02 and is restricted to the interval $[0, 0.10]$. When the stagnation trigger becomes active, $\sigma_t$ is updated before generating the candidate. If the previous selected point was a standard PCA-BO candidate, $\sigma_t$ is increased by 0.01, up to the maximum value. If the previous selected point was orthogonal, $\sigma_t$ is decreased by 0.01, down to the minimum value. After an orthogonal point is selected, a cooldown period is applied before another orthogonal perturbation can be used.

### 3.1.3 Algorithm Summary

Figure 1 summarizes the control flow of the Dynamic Orthogonal PCA-BO method. The visual examples are included to make the PCA direction and the orthogonal step easier to see in a simple two-dimensional case.

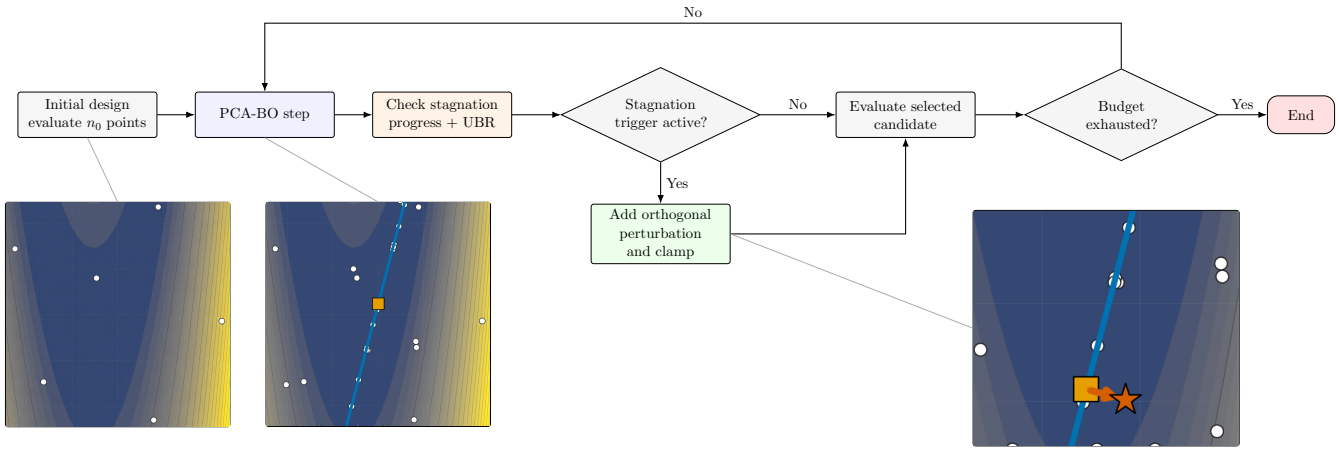


Figure 1: Flowchart of the Dynamic Orthogonal PCA-BO method. The visual examples illustrate the initial design, the PCA-BO candidate, and an orthogonally perturbed candidate in a two-dimensional case.

Algorithm 3 gives the same Dynamic Orthogonal PCA-BO procedure in pseudocode.

**Algorithm 3** Dynamic Orthogonal PCA-BO

**Require:** Objective function $f$, search domain $\mathcal{X}$, evaluation budget $N$, initial design size $n_0$

Generate initial design $X = \{x_1, \ldots, x_{n_0}\} \subset \mathcal{X}$
Evaluate $y_i = f(x_i)$ for $i = 1, \ldots, n_0$
Initialize UBR history, stagnation counter, perturbation scale $\sigma$, and cooldown counter
**for** $n = n_0, \ldots, N-1$ **do**
    Generate PCA-BO candidate $x_{\text{PCA}}$ using Algorithm 2
    Compute stagnation signal from recent best-value progress
    Compute UBR plateau signal from the current latent GP model
    **if** stagnation trigger is active, cooldown is inactive, and $r < D$ **then**
        Sample Gaussian perturbation $\epsilon$
        Project perturbation onto the PCA-orthogonal complement
        Set $x_{n+1} \leftarrow \text{clamp}(x_{\text{PCA}} + \epsilon_\perp, \mathcal{X})$
    **else**
        Set $x_{n+1} \leftarrow x_{\text{PCA}}$
    **end if**
    Evaluate $y_{n+1} \leftarrow f(x_{n+1})$
    Update $X$, $y$, best value, UBR history, perturbation scale, and cooldown counter
**end for**
**return** Best evaluated point

## 3.2 Experimental Setup

All experiments use the same benchmark protocol unless stated otherwise. Performance is reported as best-so-far loss.

Table 1: Shared experimental setup used throughout the thesis.

| Setting | Value |
|---|---|
| Benchmark suite | BBOB / IOHprofiler |
| Functions | f2, f8, f16, f19, f22 |
| Dimensions | 10, 20, 40 |
| Instances | 1, 2, 3 |
| Seeds in 10D and 20D | 12–21 per instance |
| Runs in 10D and 20D | 30 per function and method |
| Seeds in 40D | 12–16 per instance |
| Runs in 40D | 15 per function and method |
| Initial design size | $n_0 = 3D$ |
| Evaluation budget | $30D$ |
| Performance metric | Best-so-far loss, $f_{\text{best}}(t) - f^*$ |
| Reported curve | Mean over matched runs |
| Uncertainty band | 95% confidence interval |

Dynamic Orthogonal PCA-BO is compared against standard BO and PCA-BO using the same benchmark functions, dimensions, instances, seeds, initial design size, evaluation budget, and

performance metric. Standard BO is implemented with BoTorch [1], PCA-BO follows Raponi et al. [8]. The method-specific settings of Dynamic Orthogonal PCA-BO are summarized in Table 2.

Table 2: Main hyperparameters of the Dynamic Orthogonal PCA-BO method.

| Parameter | Value |
|---|---|
| PCA variance threshold | 95% |
| Stagnation patience | 10 iterations |
| Best-value improvement tolerance | $10^{-8}$ |
| UBR smoothing window | 7 |
| UBR plateau threshold | 0.1 |
| Required consecutive active checks | 2 |
| Cooldown after orthogonal step | 4 iterations |
| Initial orthogonal noise scale | 0.02 |
| Maximum orthogonal noise scale | 0.10 |
| Noise scale step | 0.01 |
| Candidate bounds | Global benchmark bounds |

## 3.3 Results

Figure 2 shows the 10D convergence behaviour of BO, PCA-BO, and Dynamic Orthogonal PCA-BO, averaged over 30 matched runs. The results are mixed. On f2 and f8, Dynamic Orthogonal PCA-BO improves over PCA-BO, but BO eventually reaches the lowest final loss. This suggests that the orthogonal step can improve the PCA-based search, although it is not enough to make it the best method on these functions.

On f16 and f19, PCA-BO remains the leading PCA-based method. Dynamic Orthogonal PCA-BO is still competitive, but the additional exploration does not improve the final results. The clearest positive case is f22, where PCA-BO stagnates early while the dynamic method continues on improving and finishes close to BO. Overall, the 10D results show that the dynamic step can help, but its effect is function-dependent.

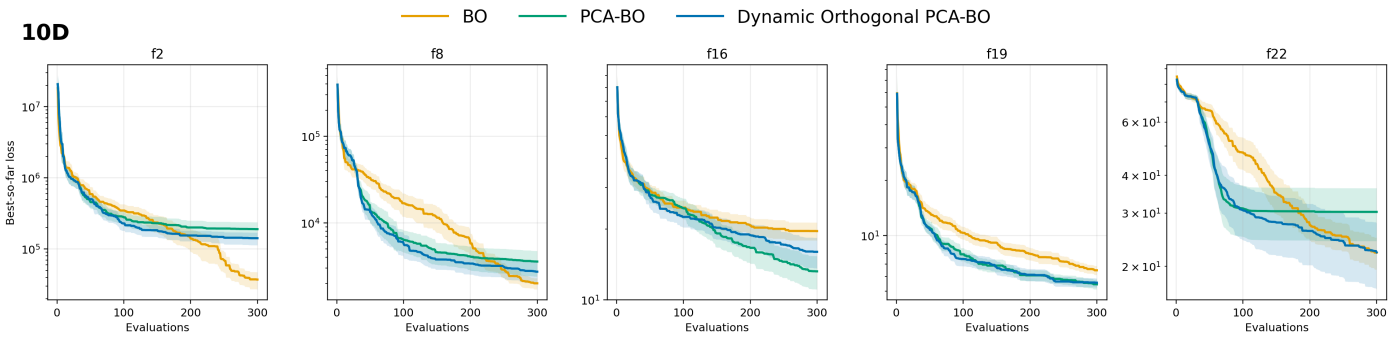


Figure 2: Convergence comparison for BO, PCA-BO, and Dynamic Orthogonal PCA-BO on the 10D benchmark functions.

Figure 3 shows the same comparison in 20D. Compared with 10D, the advantage over BO becomes even more clear: both PCA-based methods outperform BO on most functions. However,

Dynamic Orthogonal PCA-BO does not consistently outperform PCA-BO. On f2, f16, and f19, PCA-BO remains slightly stronger or very close to the dynamic variant.

The dynamic method is most useful on f8 and f22. On f8, it finishes slightly ahead of PCA-BO, while both PCA-based methods are much better than BO. On f22, the difference is much clearer: BO and PCA-BO flatten at higher loss values, while Dynamic Orthogonal PCA-BO continues improving throughout the budget. Thus, in 20D, the dynamic method gives a clear benefit on some functions, but not a uniform improvement over the PCA-BO baseline.

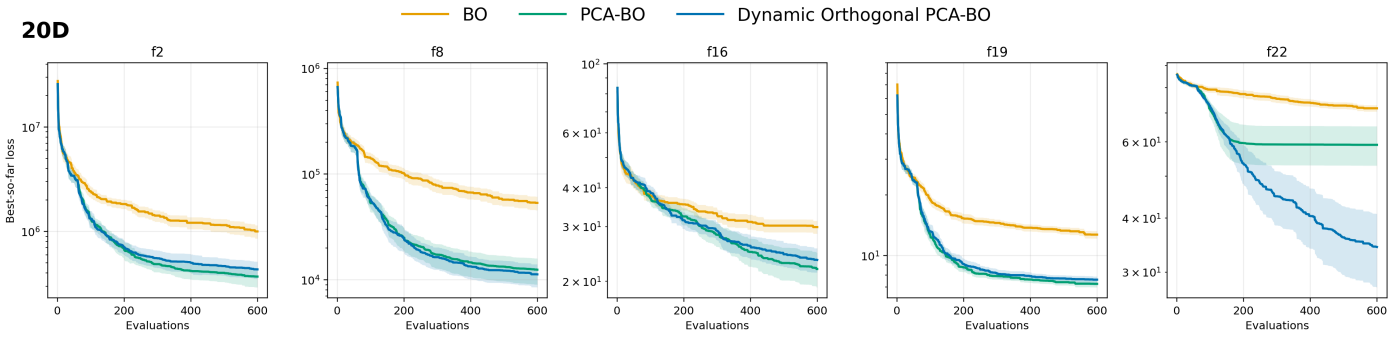


Figure 3: Convergence comparison for BO, PCA-BO, and Dynamic Orthogonal PCA-BO on the 20D benchmark functions.

Figure 4 shows the 40D comparison. The same pattern remains visible. On f2, both PCA-based methods strongly outperform BO, but PCA-BO finishes slightly better than the dynamic method. On f8, Dynamic Orthogonal PCA-BO gives a small improvement over PCA-BO. On f16 and f19, PCA-BO remains stronger, although the dynamic method still improves clearly over BO on f19.

The strongest 40D result for Dynamic Orthogonal PCA-BO is again f22. BO makes little progress after the initial phase, and PCA-BO also flattens before the dynamic method. Dynamic Orthogonal PCA-BO continues improving for longer and reaches the lowest loss among the three methods. Overall, the 40D results support the same conclusion as the lower-dimensional experiments: orthogonal exploration can be useful, especially on f22 and to a smaller extent f8, but it does not consistently dominate a strong PCA-BO baseline.

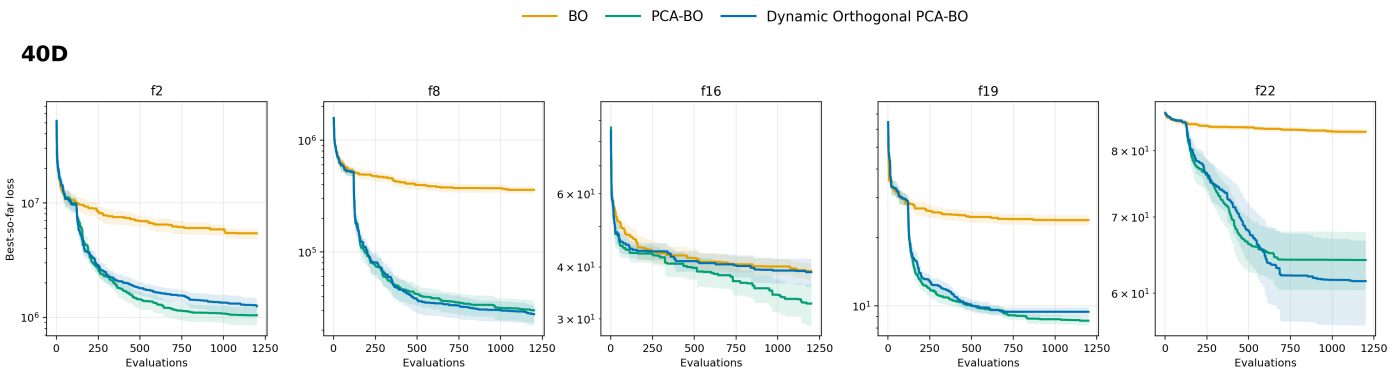


Figure 4: Convergence comparison for BO, PCA-BO, and Dynamic Orthogonal PCA-BO on the 40D benchmark functions.

## 3.4 Limitations

Figure 5 adds TuRBO-1 [3] as a stronger Bayesian Optimization reference. The comparison shows that Dynamic Orthogonal PCA-BO can improve over PCA-BO on selected functions, especially f22, but it remains far from TuRBO-1 on most benchmarks.

This gap is visible across all three dimensions. TuRBO-1 usually reduces the loss much faster and continues improving after the global PCA-based methods have started to flatten. The difference is especially clear on f2, f8, f16, and f22. On f19, the gap is smaller, but TuRBO-1 remains competitive with or better than the PCA-based methods.

These results show the main limitation of the dynamic orthogonal method. The orthogonal perturbation can add useful directions outside the current PCA subspace, but the method still uses one global PCA-BO search process. It does not adapt the size of the search region or fit the model around local structure. Therefore, the dynamic orthogonal step can reduce some stagnation, but it does not provide the same level of local adaptation as TuRBO-1.

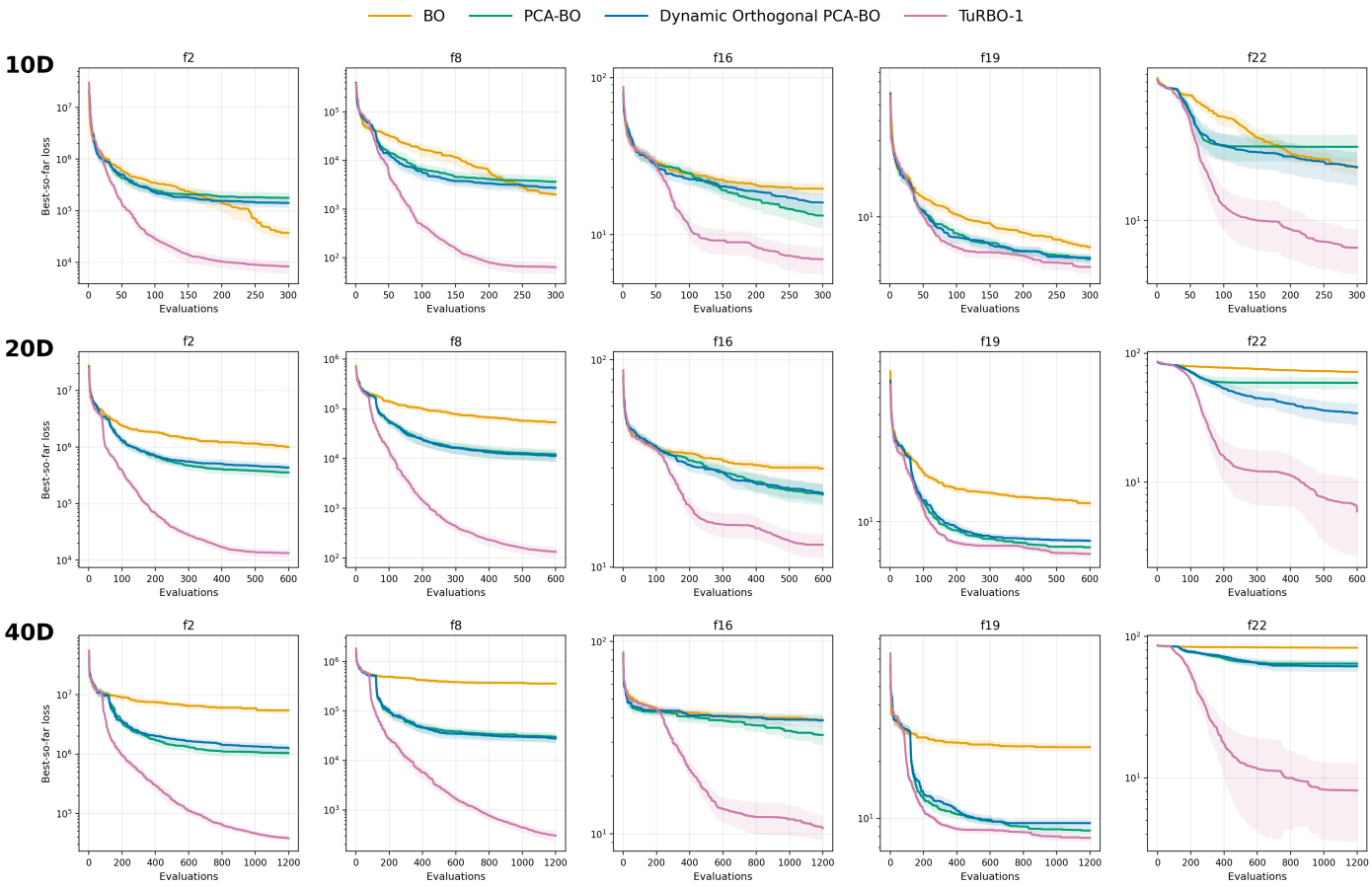


Figure 5: Convergence comparison including TuRBO-1 in 10D, 20D and 40D.

# 4 Trust-Region Extension

## 4.1 Motivation

The comparison with TuRBO-1 shows that the dynamic orthogonal method is still far behind a local Bayesian Optimization baseline. This suggests that adding orthogonal exploration to global PCA-BO is not enough by itself. The next step is therefore to study whether the same dynamic orthogonal mechanism becomes more effective when it is combined with a trust-region search structure.

## 4.2 Methodology

The trust-region extension applies Dynamic Orthogonal PCA-BO inside an adaptive local search region. This section describes only the additional trust-region components. The PCA-BO, orthogonal perturbation, and stagnation-detection logic are reused from Section 3.

### 4.2.1 Trust Region Definition

The trust region is defined in the original search space, not in the PCA latent space. This is important because the PCA subspace may change during the optimization, while the original input variables remain fixed. Defining the trust region in the original space ensures that the local search region corresponds to a real region of the benchmark function.

Following the TuRBO setup, the search domain is first scaled to the unit hypercube $[0,1]^D$, where $D$ is the original dimensionality. If the original lower and upper bounds are denoted by $l$ and $u$, a point $x$ is normalized as

$$x_{\text{unit}} = \frac{x - l}{u - l}.$$

The trust-region center, length, and bounds are stored in this normalized space. In the final implementation, one active trust region is used. The region is represented by a center point $c$ and a length $L$. The center is initialized from the best point found in the initial design, and the initial length is

$$L_{\text{init}} = 0.8.$$

The trust-region bounds are an axis-aligned box around the center,

$$c - \frac{L}{2} \quad \text{and} \quad c + \frac{L}{2},$$

clipped to remain inside $[0,1]^D$. The extension uses this uniform trust-region box. Candidate points generated by the local PCA-BO step or by the orthogonal refinement are constrained to remain inside the active trust region.

### 4.2.2 Local Dynamic Orthogonal PCA-BO

At each iteration, the method selects a training set around the trust-region center. Previously evaluated points inside the current trust-region bounds are selected first. If more than $6D$ points are available, only the $6D$ closest points to the center are kept. If fewer points are available, the set is completed using the nearest points from the full evaluation history. The weighted PCA and latent-space Gaussian Process are then fitted on this selected set.

The candidate is generated by optimizing the penalized expected improvement acquisition in the resulting PCA latent space. The trust-region center is projected into this space, and the acquisition search is restricted to a box around the projected center.

$$\rho_{\mathrm{TR}} = \frac{1}{2}\gamma \min_j \left(u_j^{\mathrm{TR}} - l_j^{\mathrm{TR}}\right), \qquad \gamma = 0.75.$$

Thus, the latent search scale follows the current trust-region size. After the best latent candidate is selected, it is mapped back to the original search space.

The stagnation detection from Section 3.1.2 is reused, with the uncertainty-bound ratio computed over the local latent bounds. If the trigger is inactive, the mapped candidate is evaluated directly as a manifold step. If the trigger is active and the local PCA rank is smaller than $D$, one Gaussian perturbation is sampled, projected onto the orthogonal complement of the local PCA subspace, and added to the mapped candidate.

The perturbation scale is also tied to the trust-region size:

$$\sigma_{\mathrm{TR},t} = \sigma_t \cdot \frac{1}{D}\sum_{j=1}^{D}\left(u_j^{\mathrm{TR}} - l_j^{\mathrm{TR}}\right) \cdot \eta_{\mathrm{TR}}, \qquad \eta_{\mathrm{TR}} = 0.25.$$

Here, $\sigma_t$ is the dynamic orthogonal noise scale from the main method. Both the manifold and orthogonal candidates are clamped to the active trust-region bounds before evaluation. Therefore, the method can explore outside the local PCA subspace, but it cannot leave the current trust region.

### 4.2.3 Trust Region Updates and Restarts

After each evaluation, the trust-region state is updated based on whether the new point improves the best value associated with the region. If the point improves the region's best value, it becomes the new trust-region center. The success counter is increased and the failure counter is reset. Otherwise, the center remains unchanged, the success counter is reset, and the failure counter is increased.

The trust-region length is then adapted using these counters. After enough consecutive successful evaluations, the region is expanded, up to the maximum allowed length. After enough consecutive unsuccessful evaluations, the region is shrunk. In simplified form, these updates are

$$L \leftarrow \min(L_{\max}, 2L)$$

for expansion, and

$$L \leftarrow \frac{L}{2}$$

for shrinking.

If the trust-region length becomes smaller than the minimum allowed length, the region is restarted. The restart does not use additional objective evaluations. Instead, Sobol points are sampled over the full normalized domain, and the new center is chosen as the Sobol point farthest from the existing evaluation history. The trust-region length and the success and failure counters are then reset.

Algorithm 4 gives the same procedure in pseudocode form.

**Algorithm 4** Trust-Region Extension

**Require:** Objective function $f$, search domain $\mathcal{X}$, evaluation budget $N$, initial design size $n_0$

Generate initial design $X = \{x_1, \ldots, x_{n_0}\} \subset \mathcal{X}$
Evaluate $y_i = f(x_i)$ for $i = 1, \ldots, n_0$
Initialize trust-region center $c^{\mathrm{TR}}$, length $L$, success counter, and failure counter
**for** $n = n_0, \ldots, N-1$ **do**
    Define trust-region bounds $\mathcal{X}^{\mathrm{TR}}$ around $c^{\mathrm{TR}}$
    Select local data $(X_{\mathrm{local}}, y_{\mathrm{local}})$ from evaluated points near $\mathcal{X}^{\mathrm{TR}}$
    Generate and evaluate $x_{n+1}$ using Algorithm 3 on $(X_{\mathrm{local}}, y_{\mathrm{local}})$ and $\mathcal{X}^{\mathrm{TR}}$
    Add $(x_{n+1}, y_{n+1})$ to the global evaluation history
    Update success or failure counter using $y_{n+1}$
    **if** success counter reaches threshold **then**
        Expand trust-region length $L$
    **else if** failure counter reaches threshold **then**
        Shrink trust-region length $L$
    **end if**
    **if** $L < L_{\min}$ **then**
        Restart trust region and reset counters
    **end if**
**end for**
**return** Best evaluated point

## 4.3 Experimental Setup

The trust-region experiments use the same benchmark setup as in the previous section, while the additional settings introduced by the trust-region extension are summarized in Table 3.

Table 3: Main settings of the trust-region extension.

| Setting | Value |
|---|---|
| Number of trust regions | 1 |
| Trust-region shape | Equal-side-length box |
| Initial trust-region length | $L_{\text{init}} = 0.8$ |
| Minimum trust-region length | $L_{\text{min}} = 2^{-7}$ |
| Maximum trust-region length | $L_{\text{max}} = 1.6$ |
| Success tolerance | 3 consecutive successful evaluations |
| Failure tolerance | $D$ consecutive unsuccessful evaluations |
| Local training set size | At most $6D$ points |
| Latent radius scale | 0.75 |
| Trust-region orthogonal scale | 0.25 |
| Restart mode | Distance-based Sobol restart |

## 4.4 Results

Figure 6 shows the 10D results after adding the Local Dynamic Orthogonal PCA-BO method. The trust-region version improves over the global dynamic method on most functions, but the improvement is not uniform. On f2, the local method improves over PCA-BO and Dynamic Orthogonal PCA-BO, although it still flattens before reaching BO or TuRBO-1. On f8, it gives the strongest result among the PCA-based methods, while TuRBO-1 remains clearly better.

The benefit of the local version is most visible on f16 and f19. On f16, Local Dynamic Orthogonal PCA-BO continues improving after the global PCA-based methods have slowed down and finishes close to TuRBO-1. On f19, the methods are closer overall, but the local method remains among the best and slightly improves over the global dynamic variant. The main exception is f22, where the local method performs worse than the global dynamic method and stays far from TuRBO-1. Overall, the 10D results show that the trust-region extension can help, but it is still function-dependent.

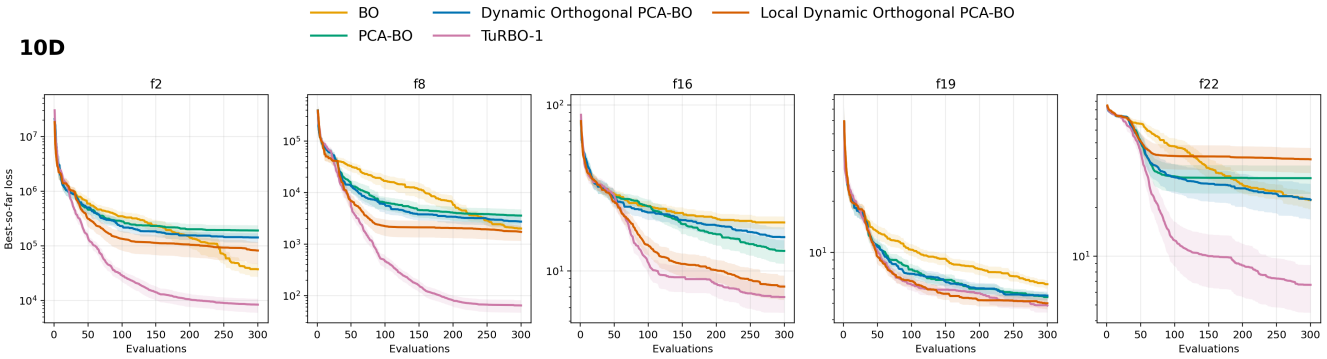


Figure 6: Convergence comparison in 10D with the Local Dynamic Orthogonal PCA-BO method included.

Figure 7 shows the same comparison in 20D. The local method performs better than the global dynamic method on most functions, making the effect of the trust-region extension clearer than in 10D. On f2 and f8, it is the strongest PCA-based variant, although TuRBO-1 still reaches a

much lower loss. On f16, Local Dynamic Orthogonal PCA-BO gives one of its strongest results and finishes close to TuRBO-1.

On f19, the methods are again closer, but the local dynamic method remains competitive and improves over the global dynamic version. The weakest case is f22, where the global dynamic method performs better than the local one, while TuRBO-1 remains clearly ahead of both. Overall, the 20D results show that using a local trust-region structure often improves the dynamic orthogonal approach, especially on f2, f8, and f16.

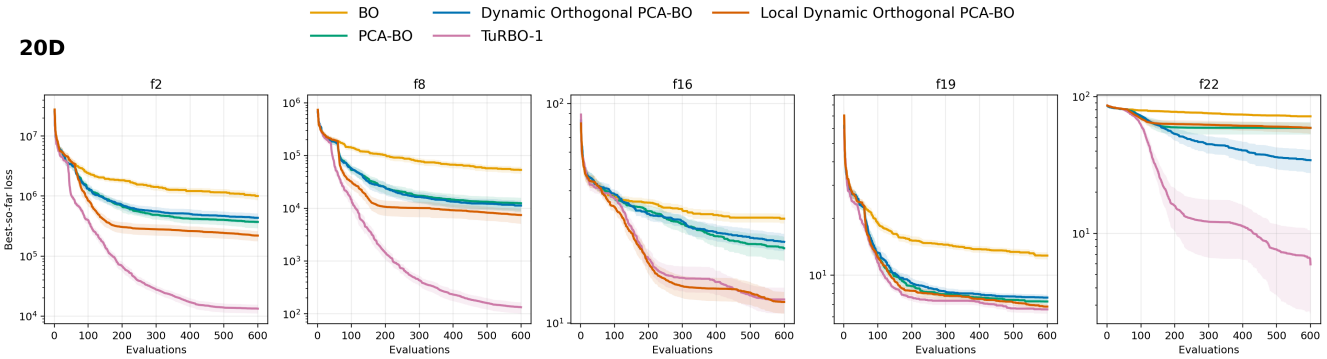


Figure 7: Convergence comparison in 20D with the Local Dynamic Orthogonal PCA-BO method included.

Figure 8 shows the comparison in 40D. The local method remains useful on f2 and f8, where it improves over the global dynamic method and both PCA-BO baselines. However, TuRBO-1 still converges substantially faster on both functions. On f16, the local method is much stronger than the global dynamic method and follows TuRBO-1 more closely, which is similar to the behaviour already seen in 10D and 20D.

On f19, the local method is again competitive with TuRBO-1 and PCA-BO, with only small differences between the best methods near the end of the budget. The main weak case remains f22. Here, the local method improves only slightly over PCA-BO and stays behind both the global dynamic method and TuRBO-1. Overall, the 40D results support the same pattern as the lower-dimensional experiments: the trust-region extension often improves the dynamic method, especially on f2, f8, f16, and f19, but it still does not match the robustness of TuRBO-1.

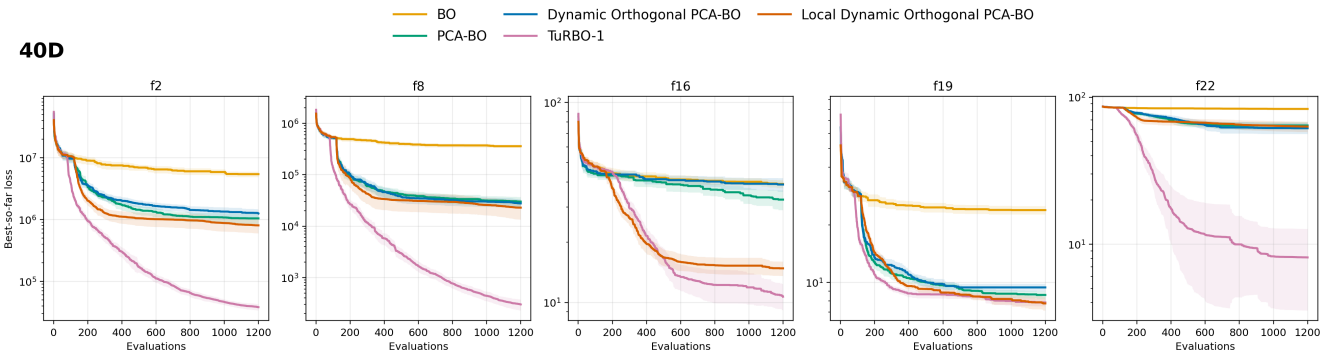


Figure 8: Convergence comparison in 40D with the Local Dynamic Orthogonal PCA-BO method included.

# 5 Discussion

## 5.1 Effect of Orthogonal Exploration

Dynamic Orthogonal PCA-BO was introduced to address one specific weakness of PCA-BO – stagnation. The results suggest that the orthogonal step can help in this situation, but its effect is function-dependent.

Our results are consistent with Raponi et al. [8], who observed that PCA-BO performs better on BBOB functions with adequate global structure. In this thesis, PCA-BO remains competitive on f2, f8, f16, and f19, and Dynamic Orthogonal PCA-BO usually performs similarly. This may suggest that, when the PCA projection already provides a good search representation, orthogonal exploration has limited room to improve the overall convergence.

The clearest contrast is f22, where Dynamic Orthogonal PCA-BO most clearly outperforms PCA-BO across all dimensions considered. This is consistent with the interpretation that f22 is harder to describe with one global PCA projection. For such functions, the latent space has a higher chance on missing important regions and excluding the directions needed for further improvement. Orthogonal exploration is useful precisely in this situation, because it tests directions that are not in the current PCA subspace.

The interpretation is supported by the statistics of dynamic exploration. The improvement on f22 is not explained by perturbations happening more often. For example, in 20D, orthogonal exploration is applied in 18.1% of decision steps on f2, 19.0% on f16, and 17.7% on f19, but only 13.3% on f22. Nevertheless, f22 is one of the clearest cases where the dynamic method outperforms PCA-BO. This suggests that the important factor is not only how often perturbations are applied, but whether they occur in parts of the search space where they can meaningfully redirect the search.

Figure 9 gives a qualitative example of such a case in two dimensions. Before evaluation 53, the PCA model is effectively using only one principal component, so the search is restricted to a single direction and therefore stagnating. At evaluation 53, the exploration trigger fires and the candidate is pushed away from its original place. The perturbed point lands in a better region and improves the best value. Once this point is added to the data, the PCA model builds a second principal component, and the following evaluations start exploring around the new region instead of continuing along a single line. This shows the exact intended behaviour of the method: an orthogonal step can push the search out of stagnation by giving PCA-BO new information that was not covered before.

## 5.2 Role of Locality

On f2 and f8, adding locality generally improves the global dynamic method. The local version reduces the loss more quickly and often becomes the strongest PCA-based method, although TuRBO-1 still remains ahead. On f19, all the methods are much closer across dimensions. The local method remains competitive, but the curves do not separate as strongly as on the other functions, so the effect of locality is less visible there.

The strongest positive case for the local method is f16, where it performs very similarly to TuRBO-1 across all tested dimensions. This suggests that the local search technique is important for this function. A global PCA projection may average over too much of the search history, while

the trust region allows the model to focus on the part of the function currently being explored. The fact that the local method has a convergence shape similar to TuRBO-1, but shifted to worse values, suggests that the trust-region mechanism is driving much of the optimization behaviour.

The behaviour on f22 is almost the opposite. Dynamic Orthogonal PCA-BO performs well on f22, but the local trust-region version performs worse. A plausible explanation is that the trust region bounds restrict the exploration that made the global dynamic method good. In the local method, the perturbation is scaled by the trust-region size and clamped to the active region. If the region becomes too small or is centered in a bad part of the domain, the orthogonal step may no longer find a better place in the search space.

## 5.3 Limitations

A limitation of Dynamic Orthogonal PCA-BO is that the orthogonal step is only a correction to the selected candidate. It can move a point outside the current PCA subspace, but the surrogate model and the PCA projection are still built from the evaluation history. Therefore, the method can react to stagnation, but it does not fully change the global search mechanism of PCA-BO. This helps to explain why the method can improve over PCA-BO on some functions, while still remaining behind TuRBO-1 on most of the benchmarks.

The local extension reduces this limitation, but introduces another one. By constraining the search to a trust region, it can make the model more focused, but it can also reduce the exploratory freedom of the orthogonal step.

Another limitation is that the stagnation trigger and perturbation scale are controlled by fixed hyperparameters. The same patience window, UBR threshold, cooldown, and noise-scale update rule are used across functions and dimensions. The results suggest that the usefulness of orthogonal exploration depends on the function, so a fixed trigger rule may not be ideal. A more adaptive rule could potentially distinguish better between cases where the PCA subspace is genuinely restrictive and cases where the search is simply progressing slowly.

Overall, the experiments suggest that controlled orthogonal exploration can reduce some stagnation effects of PCA-BO. However, the benefit is function-dependent. Adding locality can improve the method when a focused local search is useful, but it can also restrict exploration when broader movement is needed. This means that reducing stagnation in PCA-BO is not only a question of adding more exploration, but also of adding the right kind of exploration depending on the objective function.

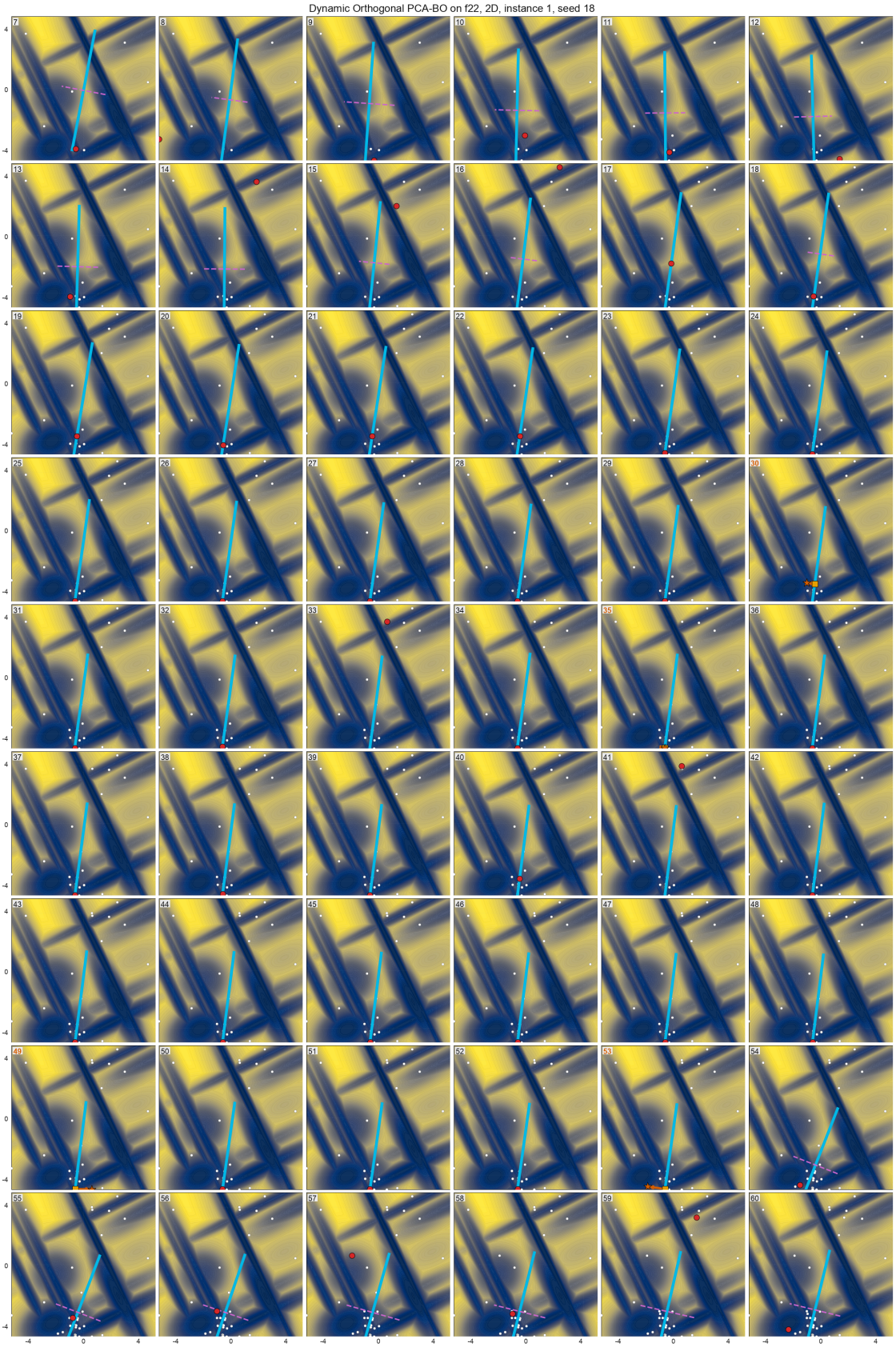


Figure 9: Iteration-level visualization of Dynamic Orthogonal PCA-BO on f22 in 2D. The panels show the evaluated points, the current PCA directions, and the orthogonal perturbation steps during one representative run.

# 6 Conclusions and Further Research

This thesis investigated whether PCA-BO can be extended to reduce stagnation in high-dimensional Bayesian Optimization. The results show that Dynamic Orthogonal PCA-BO can improve over PCA-BO on several benchmark settings, but that the benefit depends on the structure of the objective function. Orthogonal exploration is therefore not automatically beneficial: on functions where PCA-BO stagnation is less prominent, the dynamic method often performs similarly to PCA-BO.

The trigger statistics also show that performance is not explained only by how often orthogonal exploration is activated. What matters is whether the perturbation occurs in a part of the search where it can have meaningful impact on the optimization. This suggests that the timing of orthogonal exploration is more important than its frequency.

The trust-region experiments show that applying Dynamic Orthogonal PCA-BO locally can improve the method when the objective benefits from focused local search. However, locality can also restrict the broader movement that made the global dynamic method useful. Thus, local search can help reduce stagnation in some cases, but it can also limit the exploratory behaviour needed in others.

Overall, the answer to the research question is positive but limited. Controlled orthogonal exploration can reduce PCA-BO stagnation, and local modelling can improve the effectiveness of the dynamic method. However, neither extension fully closes the gap to strong local Bayesian Optimization methods such as TuRBO-1. Future work should focus on more adaptive trigger rules, perturbation scaling, and mechanisms for deciding when the search should remain global or become local.

# References


[1] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[2] C. Benjamins, E. Raponi, A. Jankovic, C. Doerr, and M. Lindauer. Self-adjusting weighted expected improvement for bayesian optimization. In *Proceedings of the Second International Conference on Automated Machine Learning*, volume 224 of *Proceedings of Machine Learning Research*, pages 6/1–50. PMLR, 2023.

[3] 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*, 32, 2019.

[4] P. I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[5] A. Greganova. Enhancing principal component analysis-assisted bayesian optimization through local embeddings. Bachelor thesis, Leiden Institute of Advanced Computer Science, Leiden University, 2025.

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

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

[8] E. Raponi, H. Wang, M. Bujny, S. Boria, and C. Doerr. High dimensional bayesian optimization assisted by principal component analysis. In *International Conference on Parallel Problem Solving from Nature*, pages 169–183. Springer, 2020.

[9] I. M. Sobol. Distribution of points in a cube and approximate evaluation of integrals. *USSR Computational mathematics and mathematical physics*, 7:86–112, 1967.

[10] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1015–1022. Omnipress, 2010.

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