Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

# LABCAT: Locally adaptive Bayesian optimization using principal-component-aligned trust regions

by

Emile Visser

*Dissertation presented for the degree of*
*Doctor of Philosophy in Electronic Engineering in the Faculty of*
*Engineering at Stellenbosch University*

Supervisors: Dr. J.C. Schoeman
Dr. Corné E. van Daalen

March 2025

# Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: February 2025 .....

# Abstract

Bayesian optimization (BO) is a popular and well-studied technique for the optimization of black-box objective functions, using an iteratively updated Gaussian process (GP) surrogate model to approximate the objective function and inform the selection of the next point to evaluate from the objective function. BO is especially renowned for a high degree of sample efficiency, which allows it to find good solutions with relatively few objective function evaluations. However, this method has several notable shortcomings which hinder its use across a broad range of optimization problems and objective functions. Specifically, BO can experience significant computational slowdown as the number of algorithm iterations increases, which poses challenges for real-time applications. Additionally, BO may struggle with non-stationary and ill-conditioned objective functions due to the reliance on a kernel-based GP surrogate model that often requires manual kernel engineering to model these objective functions adequately. Finally, a lack of theoretical guarantees and practical, numerical limitations mean that BO often exhibits poor convergence characteristics.

Several algorithms have been proposed that incorporate local strategies into the BO framework to mitigate these limitations, such as trust regions or domain partitioning; however, none address all of them satisfactorily. To address these shortcomings, this dissertation presents the *locally adaptive Bayesian optimization using principal-component-aligned trust regions* (LABCAT) algorithm, which follows the example of other trust-region-based BO algorithms. These algorithms incorporate an iteratively resized region, known as a trust region, that is used to constrain the choice of the next sample point and, by extension, the region of the objective function being approximated by the surrogate model. Using a trust region relaxes the global focus of BO to a sequence of local optimization problems that are, ideally, easier to solve.

The proposed LABCAT algorithm extends the trust-region-based BO framework through the addition of a novel rotation which aligns the trust region with the weighted principal components of the observed data and an adaptive rescaling strategy based on the length-scales of the local GP surrogate model with automatic relevance determination. These two extensions allow better adaptation of the trust region to ill-conditioning or non-stationarity and, when combined with approximative hyperparameter estimation and observation discarding schemes, allow for improved computational and convergence performance characteristics. Through extensive numerical experiments using a set of synthetic test functions and the well-known COCO benchmarking software, the LABCAT algorithm is shown to outperform several state-of-the-art BO and other black-box optimization algorithms.

# Opsomming

Bayesiese optimering (BO) is 'n gewilde en goed-bestudeerde tegniek vir die optimering van swartboksdoelfunksies deur gebruik te maak van 'n iteratief opgedateerde Gaussiese proses (GP) surrogaatmodel om die doelfunksie te benader en die keuse van die volgende punt in die evaluering van die doelfunksie te beïnvloed. BO is veral bekend vir 'n hoë mate van evaluasiedoeltreffendheid, wat leen tot die vind van goeie oplossings met relatief min doelfunksie evaluerings. Hierdie metode het wel verskeie terkortkominge wat die gebruik daarvan belemmer vir 'n wye groep van optimeringsprobleme en -doelfunksies. Eerstens kan BO aansienlike berekeningsvertraging ervaar soos die aantal iterasies van die algoritme vermeerder, wat uitdagings bied vir intydse toepassings. Bonop kan BO sukkel met doelfunksies wat nie stasionêr is nie of wat sleg gekondisioneer is as gevolg van die afhanklikheid van 'n kernfunksiegebaseerde GP wat dikwels handgedrewe kernfunksieverstelling benodig om hierdie tipes doelfunksies tot 'n voldoende mate te modelleer. Uiteindelik veroorsaak 'n gebrek aan teoretiese waarborge en praktiese, numeriese beperkings dat BO dikwels swak konvergensie-eienskappe vertoon.

Alhoewel verskeie algoritmes voorgestel is wat plaaslike strategieë kombineer met die BO raamwerk om hierdie terkortkominge aan te spreek, soos vertroue streke of domeinondervedeling, spreek geen een van hulle volledig die terkortkominge aan nie. Om die terkortkominge aan te spreek stel hierdie proefskrif die "*locally adaptive Bayesian optimization using principal-component-aligned trust regions* (LABCAT)" algoritme voor, wat die voorbeeld volg van ander vetroustreeksgebaseerde algoritmes. Hierdie algoritmes inkorporeer 'n iteratief aangepasde area, wat bekend staan as 'n vertroue streek, om die keuse van die volgende evaluasiepunt te beperk en, deur uitbreiding, die area van die doelfunksie wat benader word deur die surrogaatmodel te beperk. Die gebruik van 'n vertroue streek verslap die globale fokus van standaard BO na 'n reeks van lokale optimeringsprobleme wat ideaal makliker is om op te los.

Die voorgestelde LABCAT algoritme brei die vetroustreeksgebaseerde BO raamwerk uit deur middel van 'n nuwe rotasie wat die vertroue streek belyn met die geweegde hoofkomponente van die waargenome data en 'n aanpasbare herskaleeringstrategie gebaseer op die lengteskaal van die lokale GP surrogaatmodel met automatiese relevansiebepaling. Hierdie twee uitbeidings veroorsak beter aanpassing van die vertroue streek net die nie-stasionariteit of swak konditionering en, gekombineer met benaderde hiperparameterskatting- en waarnemingsverwyderingskema, laat verbeterde berekenings- en konvergensie-eienskappe toe. Deur gebruik te maak van omvattende numeriese experimente met 'n stel sintetiese doelfunksies en die bekende COCO maatstafsagteware, vertoon die LABCAT algoritme beter resultate as verskeie moderne Bayesiese- en ander swartboksoptimeringsalgoritmes.

# Acknowledgements

Firstly, I would like to extend my gratitude to the Wilhelm Frank Trust for their generous financial support throughout this dissertation. I sincerely hope that they can continue to provide the same opportunities to others that they have so kindly provided to me.

To my supervisors, I would like to express my gratitude for granting me the honour and privilege to make my contribution, however small, to the greatest collective endeavour mankind has ever known. Your support and input has been invaluable, and this project would be the lesser for want of it.

To all of my teachers, mentors and tutors, past and present: each and every one has left an indelible mark, big and small. If, indeed, I have seen further, then it is because the shoulders of the giants I am stood upon are giant indeed.

To my colleagues at the ESL, it has been an honour to have shared your struggle; the highs and lows, the half-solutions and almost-but-not-quite coffees. Fortune will surely smile upon you in all of your future ventures and I would not be surprised if our paths cross again.

To my friends and family, I hope this work can, in some way, justify your resolute support and belief that has carried (read: dragged) me over the finish line. *I could not do this without you.*

# Contents

# List of Figures

# List of Tables

# Nomenclature

## Acronyms and Abbreviations

| | |
|---|---|
| ARD | automatic relevance determination |
| BADS | Bayesian adaptive direct search |
| BBOB | black-box optimization benchmark |
| BO | Bayesian optimization |
| COCO | comparing continuous optimizers |
| DFO | derivative-free optimization |
| DoE | design of experiment |
| ECDF | empirical cumulative distribution function |
| EGO | efficient global optimization |
| EI | expected improvement |
| GP | Gaussian process |
| MAP | maximum a posteriori |
| MLE | maximum likelihood estimation |
| PCA | principal component analysis |
| SAEA | surrogate assisted evolutionary algorithm |
| SMBO | sequential model-based optimization |
| SoD | subset of data |
| SoR | subset of regressors |
| SRSM | successive response surface method |
| SVD | singular value decomposition |
| TREGO | trust region efficient global optimization |
| TRIKE | trust region implementation in Kriging-based optimization with expected improvement |
| TRLBO | trust region based local Bayesian optimization |
| TuRBO | trust region Bayesian optimization |

## Symbol Conventions

| | |
|---|---|
| $\mathbf{x}$ | vector |
| $\mathbf{X}$ | matrix |
| $f(x)$ | scalar function; $f : \mathbb{R} \to \mathbb{R}$ |
| $f(\mathbf{x})$ | function; $f : \mathbb{R}^m \to \mathbb{R}$ |
| $x \mapsto x^2$ | anonymous function; element $x$ maps to $x^2$ |
| $X$ | set |
| $\{x \mid y(x)\}$ | set with elements $x$ such that predicate $y(x)$ holds |
| $X \setminus Y$ | set difference; $X \setminus Y = \{x \in X \mid x \notin Y\}$ |
| $\min X$ | minimal element of the set $X$ |
| $\operatorname{argmin}_{x \in S} f(x)$ | argument of the minimum for the function $f$ over the domain $S$; $\{x \in S \mid f(x) \leq f(s)\, \forall s \in S\}$ |
| $X \to Y$ | map; transformation from $X$ to $Y$ |
| $X \leftrightarrow Y$ | bijection; invertible transformation from $X$ to $Y$ |
| $\lvert\mathbf{X}\rvert, \lvert X\rvert$ | determinant of matrix $\mathbf{X}$ or cardinality of set $X$ |
| $\lVert\mathbf{X}\rVert_F$ | Frobenius norm of matrix $\mathbf{X}$ |
| $\mathbf{X}^\top$ | transpose of matrix $\mathbf{X}$ |
| $\mathbf{X}^{-1}$ | inverse of matrix $\mathbf{X}$ |
| $\mathbf{X}^{\frac{1}{2}}$ | square root of matrix $\mathbf{X}$ |
| $\mathbf{X}^{-\frac{1}{2}}$ | inverse of matrix square root $\mathbf{X}^{\frac{1}{2}}$ |
| $x'$ | linearly transformed scalar $x$; $\mathbb{R} \to \mathbb{R}$ |
| $\mathbf{x}'$ | linearly transformed vector $\mathbf{x} \in \mathbb{R}^m$; $\mathbb{R}^m \to \mathbb{R}^m$ |
| $\mathbf{X}'$ | linearly transformed matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$; $\mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$ |
| $X'$ | linearly transformed set $X$; $\lvert X\rvert = \lvert X'\rvert$ |
| $\ln(x)$ | natural logarithm |
| $\exp(x)$ | exponential function |
| $\operatorname{tr}(\mathbf{X})$ | trace of matrix $\mathbf{X}$ |
| $\operatorname{diag}(x, y, z)$ | diagonal matrix with elements $x$, $y$ and $z$ |
| $p(x)$ | probability distribution over $x$ |
| $p(x \mid y)$ | conditional probability distribution over $x$ given $y$ |
| $\mathbb{E}_x[f(x)]$ | expectation of $f(x)$ with respect to $x$ |
| $\operatorname{var}(x)$ | variance of variable $x$ |
| $\operatorname{cov}(x, y)$ | covariance of variable $x$ and $y$ |
| $\operatorname{std}(x)$ | standard deviation of variable $x$ |
| $:=$ | defined as |
| $\sim$ | distributed according to |
| $\propto$ | proportional to |
| $\in$ | element of |
| $\notin$ | not element of |
| $\subset$ | subset of |
| $\forall$ | for all |
| $\exists$ | there exists |

## Additional Subscript Conventions

| | |
|---|---|
| $x_i$ | entry of vector $\mathbf{x}$ at $i$-th position |
| $X_{ij}$ | entry of matrix $\mathbf{X}$ at $i$-th row and $j$-th column |
| $\mathbf{x}_i, y_i$ | $i$th observed input and output |
| $\mathbf{x}_{\min}, y_{\min}$ | minimum observed input and output |
| $\mathbf{x}_{\max}, y_{\max}$ | maximum observed input and output |
| $x_0$ | value of variable $x$ at algorithm initialization |
| $x_{t-1}$, $x_t$, $x_{t+1}$ | value of variable $x$ at the previous, current and next algorithm iteration, respectively |
| $X_{\odot}$, $\mathbf{X}_{\odot}$ | set $X$ and matrix $\mathbf{X}$ centred on $\mathbf{x}_{\min}$ |
| $X_{\circlearrowleft}$, $\mathbf{X}_{\circlearrowleft}$ | set $X$ and matrix $\mathbf{X}$ rotated according to weighted principal components |
| $X_{\leftrightarrow}$, $\mathbf{X}_{\leftrightarrow}$ | set $X$ and matrix $\mathbf{X}$ rescaled using most likely length-scales |

It is important to note that these subscripts are not necessarily mutually exclusive and, in this dissertation, additional subscripts are separated by a comma. For example, the composite symbol $\mathbf{X}'_{\circlearrowleft, t-1}$ should be read as indicating the value of some matrix $\mathbf{X}$ that is under a linear transformation ($\mathbf{X}'$) rotated according to weighted principal components ($\mathbf{X}'_{\circlearrowleft}$) from the previous algorithm iteration ($\mathbf{X}'_{\circlearrowleft, t-1}$).

## Important Symbols

| | |
|---|---|
| $\mathbf{I}$ | identity matrix |
| $\mathbf{0}_n$, $\mathbf{1}_n$ | vectors of zeroes and ones, respectively, of length $n$ |
| $\mathbb{R}$ | set of all real numbers |
| $\mathbb{R}^m$ | set of all real vectors with $m$ rows |
| $\mathbb{R}^{m\times n}$ | set of all real matrices with $m$ rows and $n$ columns |
| $\{1,\ldots,n\}$ | set of natural numbers from 1 to $n$ |
| $[a,b]^n$ | $n$-ary Cartesian product of the closed, real interval $[a,b]$, hypercube if $a=-b$ else hyperrectangle if $a\neq -b$; $[a,b]^n=\{(x_1,x_2,\ldots,x_n)\,\vert\,x_i\in[a,b]\,\forall i\in\{1,\ldots,n\}\}\subset\mathbb{R}^n$ |
| $\delta_{ij}$ | Kronecker delta function where $\delta_{ij}=1$ if $i=j$ and 0 otherwise |
| $f$ | objective function; $f:\mathbb{R}^d\to\mathbb{R}$ |
| $f_{\min}$ | global minimum of objective function |
| $d$ | dimension of objective function input space |
| $n$ | current number of objective function observations |
| $\Omega$ | set of input locations of objective function that satisfies constraints |
| $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ | Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ |
| $\mathcal{GP}(m(\cdot),k(\cdot,\cdot);\mathcal{D})$ | Gaussian process with mean function $m$ and kernel function $k$, conditioned on the set of observations $\mathcal{D}$ |
| $k(\mathbf{x}_i,\mathbf{x}_j)$ | kernel function evaluated at $\mathbf{x}_i$ and $\mathbf{x}_j$; $k:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ |
| $k_{\text{SE}}$ | squared exponential kernel |
| $\mathbf{K}$ | kernel matrix; $\mathbf{K}=[k(\mathbf{x}_i,\mathbf{x}_j)]_{1\leq i,j\leq n}$ |
| $\alpha(\mathbf{x}_*;\mathcal{D})$ | acquisition function evaluated at $\mathbf{x}_*$ given observations $\mathcal{D}$ |
| $\alpha_{\text{EI}}$ | expected improvement acquisition function |
| $m$ | Gaussian process mean function; $m:\mathbb{R}^d\to\mathbb{R}$ |
| $\boldsymbol{\theta}$ | kernel function hyperparameters |
| $\hat{\boldsymbol{\theta}}$ | MAP estimate of kernel function hyperparameters |
| $\sigma_f^2$ | signal variance |
| $\sigma_n^2$ | noise variance |
| $\ell$ | kernel length-scale |
| $\boldsymbol{\ell}$ | ARD kernel length-scales |
| $\hat{\boldsymbol{\ell}}$ | MAP estimate of ARD kernel length-scales |
| $\beta$ | trust region size factor |
| $\rho$ | cache size factor |
| $\sigma_{\boldsymbol{\ell}}$ | length-scale prior standard deviation |
| $\phi(z)$, $\Phi(z)$ | univariate standard normal probability density and cumulative distribution functions |
| $\mathcal{D}$ | set of observations as tuples; $\mathcal{D}=\{(\mathbf{x}_i,y_i)\,\vert\,i\in 1,\ldots,n\}$ |
| $X$ | set of observed inputs; $X=\{\mathbf{x}_i\,\vert(\mathbf{x}_i,y_i)\in\mathcal{D}\}$ |
| $Y$ | set of observed outputs; $Y=\{\mathbf{y}_i\,\vert(\mathbf{x}_i,y_i)\in\mathcal{D}\}$ |
| $\mathbf{W}$ | sample-wise diagonal weight matrix |
| $\mathbf{R}$ | rotation matrix |
| $\mathbf{S}$ | diagonal scaling matrix |
| $\mathbf{c}$ | offset vector |

# Chapter 1

# Introduction

*A beginning is the time for taking the most delicate care that the balances are correct. This every sister of the Bene Gesserit knows.*

— from *"Manual of Muad'Dib"* by the Princess Irulan, *Dune*

Optimization is a fundamental concept instinctively understood by humans and in nature. In general, optimization can be described as a process aimed at finding a set of inputs to a problem that achieves an outcome that is (in some sense) the most favourable, referred to as the *optimal solution*. Spanning a multitude of fields, from engineering to economics, optimization has been the subject of extensive study and widely applied. Indeed, one could argue that the question of determining the optimal distribution of limited resources in a society has been a driving force of history and development since the dawn of mankind.

The mathematical foundations of optimization can be found in the 17th century formalizations of Fermat [1] and early algorithms such as Newton's method [2] (also known as the Newton-Raphson method), with roots in ancient methods such as the Babylonian method (c. 1800–1600 BC) and Heron's method (c. 50 AD) [3]. With the advent of modern computing, development in this field accelerated with the introduction of efficient and large-scale numerical methods, shifting the field from its geometric and algebraic origins to sophisticated mathematical theory and computational methodologies.

Using standard mathematical terminology, the mathematical construct defining the relationship between the inputs and outputs of the problem to be optimized is known as the *objective function*,[1] with known (or observed) pairs of inputs and outputs known as *candidate solutions*. Optimization problems can generally be divided based on whether the inputs of the objective function are discrete or continuous, with the latter being the focus of this dissertation. These input variables (also known as decision or design variables) may also be constrained to define a space of feasible inputs in which an optimal input point, or *optimum*, must be found, with the optima often defined to be the maxima or minima of the objective function. An optimum may also be a *local* optimum if it is at least as good as any nearby inputs or a *global* optimum, which is at least as good as any other feasible input point.

Apart from the output of the objective function for a given input, additional information

[1] In various fields, the objective function is also known as the cost-, loss- or criterion function and fitness-, utility- or reward function during minimization and maximization, respectively.

may also be available or gleaned from an analysis of the objective function. This additional and meta-information is often used to guide the choice of optimization algorithm. Examples of this additional information can be found in a derivation of the objective function gradient and Hessian, or guarantees regarding the convexity, bounds and smoothness of the objective function. However, for a large subset of optimization problems, this additional information is not available and, as such, no assumptions regarding the objective function can be made. These are known as *black-box* problems, as the inner mechanism of the objective function is treated as an unknown and only the output for a given input can be observed. Black-box optimization problems often arise if the objective function lacks a closed-form expression, if the dynamics of the objective function cannot be modelled, or are not accessible.

In addition to the lack of smoothness or convexity guarantees, black-box objective functions are often not very well-behaved, such as being *non-stationary*, where the objective function exhibits different characteristics in different regions, or *ill-conditioned*, where the objective function may be orders of magnitude more sensitive to one input variable than another. In practice, these black-box functions are also often expensive to evaluate due to the computational cost, such as during the expensive simulations for aerodynamic wing design [4] and hyperparameter tuning for machine learning models [5, 6], or by being results of physical experiments, such as robot gait optimization [7]. For online and real-time systems, the underlying dynamics of the black-box function being optimized may also change with time, further necessitating an algorithm that finds a solution with minimal objective function evaluations.

In the most general form of an optimization algorithm, an initial input point is chosen with new candidate solutions that are then iteratively sampled and evaluated. Ideally, this yields a sequence of improving candidate solutions that approach an optimum of the objective function until some termination criterion (such as a desired accuracy or maximum objective function evaluation limit) is met, with different algorithms generating different sequences of candidate solutions. Naturally, some optimization algorithms, or groups thereof, may be considered as having better performance according to some metric such as accuracy, real execution time or sample efficiency for certain types of objective functions.

One popular class of optimization methods, also often the first recourse, is known as gradient-based optimization methods, such as the venerable BFGS [8] and SGD [9] algorithms. These algorithms leverage gradient information of the objective function to determine optimal step size for moving from the current candidate solution (also known as the current iterate) to the next candidate solution, improving and converging to the optimum with subsequent iterations. Gradient-based methods have been widely studied [10] and have proven to be very efficient when applied to high-dimensional and constrained problems. Often, these methods have strong convergence guarantees under certain smoothness and convexity assumptions, such as the superlinear convergence rate of BFGS [11] and quadratic rate of Newton's method [12, Ch. 9], and are near optimal choices for smooth, strictly convex objective functions.

A closely related approach, that is in some sense dual to gradient-based methods, can be found in trust-region-based optimization methods, such as the Levenberg-Marquardt [13] and Powell's dogleg [14] algorithms. For these methods, the gradient of the objective function is used to construct a local, often quadratic, approximation of the objective function surrounding the current iterate and the next input point is chosen as the optimum of this approximation, constrained by some radius or region for which the approximation is "trusted", hence the term *trust region*. Based on the value of the next iterate, the trust region is expanded if the next iterate is deemed to be successful and contracted if not. Similarly to gradient-based methods,

trust-region-based methods often have good local convergence guarantees [15] and are often more resistant to local optima than gradient-based methods.

Unfortunately, due to the strong dependence of these classical gradient-based and trust-region-based methods on the gradient of the objective function, these methods may not be the best choice for black-box objective functions. While these methods could still be applied to these functions, often using finite-difference approximations of the gradient, they may be slow to converge if the black-box function is expensive to evaluate or may even fail to converge if noise is present in the outputs of the objective function, leading to unreliable gradient information. Furthermore, due to the lack of known a priori information of black-box objective functions stated previously, the assumptions underpinning the convergence guarantees of these methods often cannot be verified.

Alternatively, methods from the field of derivative-free optimization (DFO) [16] can be applied to the problem of black-box function optimization without requiring the gradient of the objective function. This field encompasses a variety of methods that vary significantly in their approach. Some, like random search, use simple, random sampling to explore the solution space, while others use fixed rules and heuristics, such as Nelder-Mead [17] and pattern search [18]. Others incorporate mechanisms inspired by natural processes. For instance, particle swarm optimization (PSO) [19] is based on swarm behavior, simulated annealing [20] mimics the annealing process in metallurgy, and covariance matrix adaptation evolution strategy (CMA-ES) [21] is inspired by evolutionary processes. While these methods have proven to be robust and effective when applied to optimization problems with inputs of high dimensionality, it should be noted that these DFO methods are not known to be very sample efficient. This lack of sample efficiency may be due to the fixed strategies of random- and direct search methods that cannot exploit local structure of the objective function or due to the large population of candidate solutions used by particle and evolutionary methods that are sample-intensive to fully initialize and update.

One family of DFO methods that addresses the challenge of optimizing expensive black-box functions in a sample-efficient manner is known as sequential model-based optimization (SMBO) [22, 23] methods. In contrast to traditional optimization techniques, SMBO attempts to approximate the objective function with a *surrogate model*. Each subsequent objective function evaluation is added to this surrogate model, refining the approximation. The next point at which to evaluate the objective function and add to the surrogate model is then determined by maximizing an *acquisition function* that combines exploration of the objective function and exploitation of the best candidate solution. Deciding where to evaluate the objective function according to the acquisition function, and refining the surrogate model with this result, forms the core loop of SMBO given in Algorithm 1.

The rationale underpinning SMBO lies in constructing a (relatively) cheap surrogate model to approximate the computationally expensive objective function, enabling indirect optimization through computationally inexpensive evaluations of the surrogate model. In effect, this changes the problem from optimizing an expensive objective function into the optimization of another (approximate) objective function that has a closed form and is computationally cheaper. This approach minimizes direct evaluations of the objective function, allowing sample efficiency and being computationally feasible for expensive black-box objective functions.

Of particular interest for this dissertation is the popular and pre-eminent SMBO technique known as Bayesian optimization (BO) [24] with the commonly used Gaussian process (GP) surrogate model (also known as efficient global optimization (EGO) [25] and sequential Kriging

**Algorithm 1** Sequential model-based optimization (SMBO)

**Input:** Objective function, Surrogate model, Acquisition function

1: Initialize surrogate model
2: **while not** convergence criterion satisfied **do**
3: Select next input using acquisition function
4: Evaluate selected input using objective function
5: Update surrogate model
6: **end while**
7: **return** best solution found

optimization (SKO) [26]). The distinguishing feature and advantage of BO, compared to other SMBO methods, is the use of a probabilistic surrogate model that provides both an approximation of the objective function as well as the uncertainty of this approximation. This probabilistic surrogate model allows the direct use of effective acquisition functions that balance exploration of the objective function (sampling in regions of the surrogate model with high prediction uncertainty) and exploitation of the current candidate solution (sampling in regions of the surrogate model predicted to be better than the current candidate). BO has also been extensively studied [24] and applied to a wide range of real-world problems [27, App. D] from hyperparameter optimization for machine learning models [5, 6] to materials and chemical design [28, 29].

## 1.1 Research Motivation

For all of its strengths, BO is not a panacea and has shortcomings to be aware of that may limit its applicability and performance for general black-box optimization problems. Firstly, it is well-known that BO scales poorly as more observed points are added to the surrogate model (typically in the order $O(n^3)$ [30, Ch. 6] with $n$ observed points). Practically, this limits BO to lower-dimensional problems since a large number of observed points are needed to model high-dimensional objective functions, leading to computationally expensive calculations. Importantly, this may also lead to BO slowing down significantly with longer optimization runs as more observations are added to the surrogate model [31]. This slowdown can limit the applicability of BO for real-time problems where execution time guarantees are required. Sparse approximations, such as the subset of data (SoD) or subset of regressors (SoR) approaches [32], may alleviate this somewhat at the cost of surrogate model fidelity.

The second limitation of BO is that the performance of BO when applied to a specific objective function is dependent on the chosen kernel function, which is a function that defines the family of functions that the GP surrogate model is able to represent. This kernel function is normally chosen through a process known as kernel engineering, which takes prior knowledge of the objective function into account. Choosing a single, generic kernel (as is done in the case of a black-box function where there is no prior knowledge of the function) reduces the effectiveness of BO in most situations [33]. This is especially the case where the objective function is non-stationary or ill-conditioned, where the generic kernel struggles to model the wide range of behaviour and sensitivities for these functions.

Finally, the theoretical guarantees for and practical convergence characteristics of BO also

leave much to be desired. While theoretical convergence for the specific case of BO using the expected improvement acquisition function was established by Vasquez and Bect [34], explicit convergence rates depend on strong assumptions on the objective function and exist only for certain kernel functions using fixed hyperparameters, such as the work of Bull [35] or Srinivas et al. [36]. Bull also showed that for BO using sequentially estimated hyperparameters (a common approach used during BO of black-box functions), it may not converge at all. Even when these assumptions and criteria for theoretical convergence are met, BO also exhibits numerical limitations in practice, inhibiting its convergence characteristics. Computational instability in the GP model construction procedure arises when there is a close proximity between any pair of observed points in the input space, resulting in a near-singular spatial covariance matrix $\mathbf{K}$. To address this instability, a common solution is to introduce a small "nugget" parameter $\delta$ as diagonal noise ($\mathbf{K}$ is replaced by $\mathbf{K} + \delta\mathbf{I}$ [37, 38]). This reduces the rate and limit of convergence as an artificial level of noise has been implicitly imposed on the (possibly originally noiseless) objective function [39]. The previously noted computational slowdown of BO may also make convergence to an arbitrary precision impractical, as it may require many more expensive samples to reach a target $\epsilon$.

Several modified BO methods have been proposed that attempt to address one or more of the noted shortcomings of BO (with unmodified BO hereafter referred to as *standard* BO) by integrating a measure of local focus into the standard BO loop (as discussed in Chapter 2). One such modification consists of adding an iteratively updated trust region to constrain the acquisition function and, by extension, constrain the selection of the next candidate solution (hereafter referred to as *trust-region-based* BO). Unfortunately, none of these modified BO methods currently fully address all of the noted shortcomings.

## 1.2 Research Aim and Objectives

The aim of this research is to develop a modified Bayesian optimization algorithm that solves the noted shortcomings of standard BO for continuous, noiseless, black-box objective functions. In other words, an algorithm that retains the sample efficiency of standard BO while also being (i) resistant to computational slowdown, (ii) adaptable to non-stationary and ill-conditioned functions without kernel engineering, and (iii) exhibiting good convergence characteristics. Concretely, the following research objectives are identified:

- To investigate other existing modified BO methods and identify a broad strategy to form the basis of the proposed algorithm.
- To derive a novel, modified BO algorithm that addresses all of the noted shortcomings of standard BO.
- To verify the extent to which the shortcomings of BO are addressed and analyze the relative performance of the proposed algorithm compared to state-of-the-art black-box optimization algorithms using extensive and representative numerical benchmarks.

If successful, this would yield an algorithm that addresses the noted shortcomings of BO and, by addressing these shortcomings that hinder the performance of BO, may be competitive with state-of-the-art black-box optimization algorithms.

## 1.3 Solution Overview and Contributions

This research presents two novel extensions for trust-region-based BO that are combined to yield the proposed algorithm. Firstly, a strategy to adaptively rescale the trust region and objective function observations is introduced. This rescaling is based on the length-scales of the local GP surrogate model with a squared exponential kernel and automatic relevance determination, instead of a heuristic, to allow for improved convergence characteristics. The second extension is a novel rotation of the trust region to align with the weighted principal components of the observed data. This rotation enables the maximum expressive power of the automatic relevance determination kernel to model non-stationary and ill-conditioned objective functions. These two extensions are combined in a trust-region-based BO framework with an iterative, approximate hyperparameter estimation approach and a scheme that greedily discards observations to mitigate computational slowdown. This novel method is denominated as the *locally adaptive Bayesian optimization using principal-component-aligned trust regions* (LABCAT) algorithm.

Using a diverse set of synthetic test functions, a comparison of the proposed LABCAT algorithm with standard BO and a variety of trust-region-based BO algorithms shows that the LABCAT algorithm is capable of convergence to a much higher level of precision without encountering numerical issues or instability and without experiencing significant computational slowdown. A second comparison with a range of state-of-the-art black-box optimization methods from the wider field of black-box optimization, performed using the COCO benchmarking software [40], shows that the LABCAT algorithm is a strong contender in the domain of expensive black-box function optimization, significantly outperforming standard BO for nearly all tested scenarios and demonstrating exceptional performance compared to state-of-the-art black-box optimization methods, particularly in the domain of unimodal- and highly conditioned objective functions not typically associated with BO.

## 1.4 Document Outline

The remainder of this dissertation is structured as follows: Firstly, Bayesian optimization (BO) methods with some mechanism of injecting local focus are reviewed (Chapter 2), with specific interest in trust-region-based BO methods that form the basis of our proposed algorithm. Next, the prerequisite theoretical frameworks used in the contributions of this dissertation are presented. Specifically, an exploration of the Gaussian process (GP) surrogate model used in BO (Chapter 3) is provided as well as an examination of the core BO loop (Chapter 4) and the extension of BO using a trust region. Furthermore, a brief overview of the linear algebra required to calculate the weighted principal components of a set of observations is given (Chapter 5). After establishing these prerequisites, a brief overview of the proposed LABCAT algorithm is given (Chapter 6) before separate treatments of the proposed weighted-principal-component-based rotation (Chapter 7), length-scale-based rescaling (Chapter 8), and detailed description of the synthesis and implementation of the LABCAT algorithm (Chapter 9). Using this proposed algorithm, extensive tests using numerical benchmarks are performed and compared against state-of-the-art black-box optimization algorithms (Chapter 10) before final remarks and suggestions of possible avenues of future research are provided (Chapter 11).

# Chapter 2

# Review of Bayesian Optimization Methods with Local Focus

As a result of the recent popularity and research interest in Bayesian optimization (BO), every part of the BO loop—building the surrogate model, maximizing the acquisition function, and incorporating new observations—has been subject to modification. In this dissertation, we will focus on a recent avenue of research that aims to mitigate the shortcomings of BO noted in Section 1.1 by introducing a form of local focus, relaxing the global perspective of the standard BO surrogate model. The goal of these methods is to allow BO to leverage global information about the objective function to guide the search toward the optimum and then efficiently, locally exploit this solution. Each section in this chapter outlines the approach of one of several broad classes of modified BO methods and to what extent each of these classes mitigate the noted shortcomings of BO from Section 1.1. Additionally, several implementations of trust-region-based BO, the foundation chosen for the proposed LABCAT algorithm after identifying this approach as having the most potential to address the noted shortcomings of standard BO, are discussed and evaluated.

## 2.1 Hybrid Bayesian Optimization

The first class of modified algorithms consists of *hybrid* BO algorithms that add some mechanism to BO such that a switch is made to another optimization method with better convergence characteristics at some point during the execution of the algorithm to exploit the best candidate solution. This switch point may be determined using a metric such as expected gain [31], estimated regret [39] or according to a heuristic [41]. Unfortunately, determining the optimal switching point is an optimization problem in itself; switching too early or late can easily magnify the noted shortcomings of BO while reducing the sample efficiency gains of using BO in the first place. If the surrogate model of BO does not model the objective function well, such as for non-stationary or ill-conditioned functions, the determination of the switching point may also become misinformed, reducing performance.

## 2.2 Domain Partitioned Bayesian Optimization

Another class of algorithms combines BO with *domain partitioning*, which refers to a partition of the full input space of the objective function into subdivisions. With these subdivisions of the objective function, which may be ranked based on the value of the acquisition function at the centre of the subdivision [42], promising areas can be exploited and subdivided while others can be ignored in a branch-and-bound fashion. While these methods may have polynomial [43] or even exponential [42] convergence guarantees due to the efficient exploitation of the current candidate solution though the subdivision procedure, they require manual kernel engineering to adequately model the objective function and scale poorly with more observed points, similarly to standard BO.

## 2.3 Combined Local and Global Kernel Functions

A different category of methods consists of using a combined *local and global kernel function* in the surrogate model. The mechanism underpinning these methods is that a local kernel (a function defining the shape and uncertainty of the constructed surrogate model) can be used to model local changes in the objective function while a global kernel can model the wider structure. These kernels may be combined similarly to the piecewise-defined kernel of Wabersich and Toussaint [44] or the weighted linear kernel combination of Martinez-Cantin [33]. Given that standard BO is known to over-explore the search space when the kernel function of the surrogate has a high uncertainty, this drawback can be repurposed to act as a guide by allowing the local kernel a higher uncertainty than the global kernel. While being superior to standard BO when applied to non-stationary problems [33], as the local and global kernels can model differing behaviour, these combined kernel methods still scale similarly to standard BO and suffer from the same numerical issues as standard BO, inhibiting convergence to an arbitrary precision.

## 2.4 Surrogate Assisted Evolutionary Algorithms

*Surrogate assisted evolutionary algorithms* (SAEAs) [45] are a class of methods that have been inspired by the well-established field of evolutionary optimization (EO). SAEAs combine a surrogate model to approximate the objective function (also known as the fitness function in EO literature) with the traditional EO selection, crossover [46] and mutation [47] procedures to iteratively update a population of candidate solutions. By removing poor candidates from the iteratively updated population, local focus is injected into the algorithm. It is important to note that while SAEAs may not be formally considered as BO methods due to lacking an explicit acquisition function, they may be considered as such in practice due to the similar, iteratively updated surrogate model and the crossover and mutation EO steps being analogous to the acquisition function found in BO. While many SAEAs have proven to be an improvement over their standard EA counterparts [48, 49], these methods tend to have relatively large population sizes that inhibit performance in lower dimensions, both due to the additional computational overhead and the longer initial selection phases to initialize these populations before more informed selections can begin.

## 2.5 Trust-region-based Bayesian Optimization

In order to encourage standard BO to exploit the local area surrounding the best candidate solution, another class of methods utilize an adaptive *trust region*. This trust region acts as a moving window to constrain the acquisition function and, by extension, limit where the next point is sampled from the objective function. This window traverses the objective function and is permitted to expand and contract as new points are observed, with this trust-region-based approach shown in Figure 2.1.

It is important to note that the trust region does not directly limit the subspace of the objective function being approximated by the Gaussian process (GP) surrogate model as observations outside of the trust region are still allowed to be incorporated in the GP surrogate. Rather, the trust region limits the acquisition function and, by extension, where the next point is observed. This indirectly limits the subspace of the objective function being modelled, as the GP is never queried outside of the trust region.

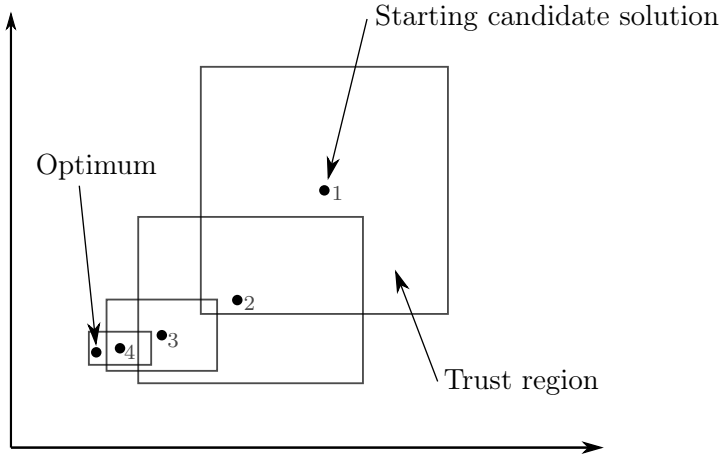



Figure 2.1: Example of Bayesian optimization of an objective function using successive trust regions, adapted from Stander and Kenneth [50]. Note the successive trust regions centred on the minimum candidate solutions and the manner in which the trust regions shrink during the approach to the optimum.

There are two distinguishing characteristics between the different trust-region-based BO algorithms to note. Firstly, and most discernibly, is the rule used for changing the size of the trust region. Some algorithms allow fine-grained control over this trust region by allowing the size of the trust region to change in each dimension independently [51] while others do so over all dimensions simultaneously [52, 53]. Secondly, some trust-region-based BO algorithms incorporate a fallback step to another algorithm [53, 54] that is interleaved with the normal BO loop and is typically taken after a certain number of steps have made no progress toward the optimum.

Incorporating a trust region relaxes the global optimization of standard BO to be more akin to that of robust local optimization, though in practice this local approach is sufficient for most problems [51]. To regain a measure of global optimization performance, trust-region-based BO methods can be paired with a complementary meta-mechanism such as alternating between a

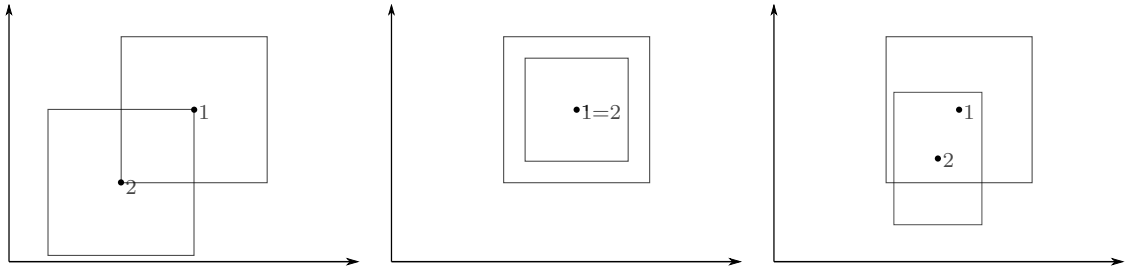
local and a global approximation [53], multistarts with a multi-armed bandit strategy [51] or restarts [52].

Due to the adaptive nature of the trust region, these trust-region-based BO methods have proven to be more resilient to non-stationary functions when compared to other methods. However, the standard use of a heuristic to update the size of the trust region precludes the use of a more informed trust region update strategy based on the local geometry of the objective function. Additionally, similar convergence characteristics as domain-partitioned BO methods may be obtained for trust-region-based BO methods through the shrinking trust region. However, this aspect has not been a primary focus in the development of existing trust-region-based BO methods, and they still face the same numerical issues encountered by standard BO methods during convergence. Similarly, the focus on the construction of a local approximation of the objective function in the trust region may also lend it itself to addressing the computational slowdown of standard BO by imposing a cap on the complexity of the local model. This aspect has also not been a focus of existing trust-region-based BO methods, which still scale similarly to standard BO methods.

The rest of this section details several prominent trust-region-based BO methods, with particular focus on the mechanism used to update the size of the trust region in each method.

### 2.5.1 SRSM

The successive response surface method (SRSM) [50] can be considered as one of the earliest trust-region-based BO methods, originally designed for use with the abstract class of response surface models, of which Gaussian processes are a member. SRSM follows the heuristic, illustrated in Figure 2.2, that if a new and better candidate solution is not found (in effect, if the trust region does not move) the trust region is made smaller ("zoom"). If a better candidate is found on the bounds of the trust region, the size of the trust region is preserved and recentred on the new candidate solution ("pan"). In the case of partial movement ("pan and zoom"), the trust region is also recentred and the trust region is made smaller by a factor determined by the axis-wise distances of the new candidate from the current one.




(a) Pan (b) Zoom (c) Pan and zoom

Figure 2.2: Illustration of the different types of trust region manipulations of the SRSM algorithm, adapted from Stander and Kenneth [50].

The SRSM also checks whether the most recently sampled point and the point before that are on the same or opposite sides of the trust region in each dimension to determine an additional contraction factor. In the case that the points are on opposite sides, it is assumed

that the trust region is oscillating around the optimum and the trust region is made smaller. For the opposite case where the two points are on the same side of the trust region, it is assumed that steady progress is being made toward the optimum and the size of the trust region is increased. Using a trust region with side lengths that can be changed independently can be advantageous for separable or ill-conditioned objective functions where the distance to the optimum along each dimension may vary greatly. However, the advantages of using independent trust region lengths depend on the surrogate model adequately approximating the objective function. Additionally, the separability or ill-conditioning of the objective function must align with the coordinate axes.

### 2.5.2 TRIKE

In contrast to the SRSM, the trust region implementation in Kriging-based optimization with expected improvement (TRIKE) [52] method uses a simplified heuristic based on the expected versus actual improvement of sampling a new point in the trust region, similar to the strategy used in the classical Levenberg-Marquardt algorithm [13]. Specifically, if the ratio of the actual improvement (the difference between the value of the next sampled point and the current candidate solution) and the expected improvement of the next sampled point exceeds a threshold, the side lengths of the trust region are increased by a constant factor. If no actual progress is made and a minimum number of observed points fall within the current trust region, the side lengths are reduced by a different constant factor. This heuristic is rather simple and, as such, is mostly outperformed by other trust-region-based BO methods.

### 2.5.3 TuRBO and TRLBO

Another method with a relatively simple trust region side length update heuristic is known as the trust region Bayesian optimization (TuRBO) method [51], as well as the subsequent trust region based local Bayesian optimization (TRLBO) method [55] based on TuRBO. The heuristic used by TuRBO and TRLBO doubles the trust region side length after a certain number of consecutive successful improvements on the current minimum candidate and halves the side length after the same number of consecutive failures.

The TuRBO and TRLBO methods also incorporate automatic relevance determination [56, 57] through an anisotropic kernel, similar to Equation 3.9 with a length-scale for each dimension. These length-scales are also used to rescale the side length of the local trust region according to the local smoothness in each dimension determined by the fitted kernel in a volume-preserving transformation. This rescaling expands the trust region in directions in which the objective function is smoother and vice versa, providing a benefit when optimizing separable objective functions. However, the automatic relevance determination employed in these techniques is limited to directions defined solely by the coordinate axes, similarly to the noted shortcoming of SRSM. As a result, TuRBO and TRLBO may not be able to exploit objective functions that are separable but not along the coordinate axes, for example, the banana-shaped valley of the Rosenbrock function [58].

Additionally, TuRBO is designed to easily be used with multiple trust regions in parallel, with objective function samples allocated to the trust region with the highest value for the acquisition function at each algorithm iteration, allowing the algorithm to be more resistant to local optima.

### 2.5.4 TREGO

The trust region efficient global optimization (TREGO) algorithm [53] comprises a trust-region-based BO algorithm interleaved with a standard BO fallback step that uses a separate global model of the objective function. This fallback step is intended to allow the trust region to escape local minima and allows for better global convergence guarantees than other trust-region-based BO methods. The trust region side length update equation for TREGO is also a success or failure heuristic that multiplies the current side lengths by constant factors. Due to TREGO maintaining both a local and global model of the objective function, it boasts impressive performance on multimodal objective functions and better convergence guarantees than other trust-region-based BO methods. However, the additional cost of maintaining these models lead to heavy computational slowdown with longer optimization runs.

### 2.5.5 BADS

Similar to the aforementioned dynamic alternation used by TREGO, Bayesian adaptive direct search (BADS) [54] switches between a trust-region-based BO method and a deterministic, model-free, direct-search fallback method. In contrast to the mixture of a local and global model used by TREGO, both of the methods used by BADS use *local* models based on trust regions. The intuition underpinning BADS is that the trust-region-based BO method can be used to make quick progress toward the optimum and when the trust-region-based BO method starts to slow down, a complementary, fail-safe, deterministic method can explore the space systematically to restart the progress of the trust-region-based BO method. The specific fallback method used by BADS is known as the mesh adaptive direct search (MADS) [59], from which the name for the BADS algorithm originates, a deterministic method that seeks to improve the current solution by testing points around the best candidate solution by moving one step in each direction on a trust-region-based mesh.

In an attempt to address the computational slowdown of standard BO, BADS adopts a subset of data (SoD) [32] observation trimming strategy to prevent the cache of observations growing infinitely. This strategy comprises preserving a minimum of 50 observed points with an additional 10 points per objective function dimension, based on the decay radius of the rational quadratic kernel used by BADS. This strategy is, however, quite conservative and still leads to the inversion of very large kernel matrices. Therefore, with longer runs, BADS still slows down considerably.

## 2.6 Evaluation of Existing Approaches

Among the different schemes proposed to mitigate the three noted standard BO shortcomings of Section 1.1 (namely experiencing computational slowdown with additional algorithm iterations, not being well-suited to non-stationary and ill-conditioned functions, and exhibiting poor convergence characteristics), none of the identified methods adequately address all of them. The trust region extension of BO is identified, however, as having the most potential to address all three of the noted shortcomings and is selected as the foundation of the proposed algorithm.

For every reviewed trust-region-based BO method, the side length of the trust region are updated using some heuristic, possibly a holdover resulting from the use of classical trust-

region-based methods transposed to the BO context. However, the information encoded in the local surrogate model of the trust region could be a valuable source for a more informed method of updating these side lengths. TuRBO and TRLBO are distinct among the identified trust-region-based BO methods due to the use of the length-scales from the surrogate model to adjust the size of the trust region. However, this adjustment is limited in two significant ways. First, the methods only perform a volume-preserving rescaling of the trust region, meaning they do not modify the trust region side lengths directly, which would change the region's volume. Second, the rescaling is constrained to align with the coordinate axes and does not allow for adjustments in arbitrary directions. These limitations mean that while TuRBO and TRLBO incorporate information from the surrogate model, their flexibility in resizing the trust region is limited.

The BADS algorithm contains an interesting SoD approach for avoiding the computational slowdown of standard BO by dropping observations from the cache that are more than a certain distance from the current minimum. This method, though, is quite conservative and better convergence characteristics may be obtained as well as the computational slowdown reduced by using a more aggressive strategy.

Finally, none of the evaluated trust-region-based BO algorithms sufficiently address the numerical issues encountered by standard BO during convergence. This is, therefore, a good avenue for improvement.

# Chapter 3

# Gaussian Processes

> *The theory of probabilities is at bottom nothing but common sense reduced to calculus; it enables us to appreciate with exactness that which accurate minds feel with a sort of instinct for which ofttimes they are unable to account.*
>
> — Pierre-Simon Laplace, *Théorie Analytique des Probabilitiés*

A core component of Bayesian optimization (BO) is the surrogate model used to construct an approximate model of the objective function using a set of observed inputs and outputs from the objective function. A popular choice for this model is the Gaussian process (GP), which builds a regression model for which unobserved points are modelled by Gaussian distributions with a predicted mean and variance. The GP is also used for the trust-region-bounded local approximation in the proposed LABCAT algorithm of Chapter 6–9. This chapter provides a review of the mechanisms needed to construct and refine such a GP model for a set of observed points. For further reading, the reader is directed to the seminal work of Rasmussen and Williams [30] or the shorter introduction to GPs by MacKay [60].

## 3.1 Gaussian Process Regression Model

The Gaussian process (GP) can be described as an extension of a multivariate Gaussian distribution to infinite dimensions [30, Ch. 1]. In other words, while multivariate Gaussian distributions describe the behaviour of a finitely long vector of a number of random *variables*, a GP describes the behaviour of a random *function*.[1] This GP model, constructed using a set of observed points $\mathcal{D} = \{(\mathbf{x}_i, y_i = f(\mathbf{x}_i)) \,|\, i \in \{1, \ldots, n\}\}$ with a chosen mean function $m(\cdot)$ and kernel function $k(\cdot, \cdot)$, can then be used as a regression model to estimate an unknown function $f(\mathbf{x})$:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot); \mathcal{D}) \tag{3.1}$$

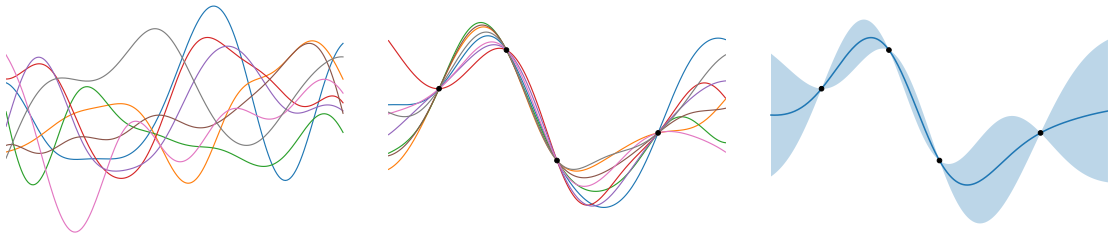
and infer predictions $y_* \in \mathbb{R}$ for unobserved input points $\mathbf{x}_* \in \mathbb{R}^d$. The key assumption of GPs is that the posterior distribution (parameterized by the predicted mean $\mu_{\mathcal{GP}}$ and variance

[1] A naive way to describe a function is to represent it as an infinitely long vector, with each entry specifying the function value $f(x)$. This analogy, while crude, is surprisingly descriptive of the mechanism of a GP.

$\sigma^2_{\mathcal{GP}}$) for these unobserved points is given by the Gaussian distribution

$$p(y_* \mid \mathbf{x}_*, \mathcal{D}) = \mathcal{N}(\mu_{\mathcal{GP}}(\mathbf{x}_*), \sigma^2_{\mathcal{GP}}(\mathbf{x}_*)), \tag{3.2}$$

with these properties illustrated in Figure 3.1.



(a) (b) (c)

Figure 3.1: Example of samples from a GP prior conditioned on observations to obtain a posterior distribution. First, (a) samples from the prior distribution of functions, with a constant prior mean and variance, are shown. Next, (b) samples from the posterior distribution of functions, conditioned on a set of observed points, are displayed, showing that the fitted functions are forced to pass through the observed points. Finally, (c) the distribution of this posterior (in effect, the GP) with the mean and standard deviation is plotted.

As seen in Equation 3.1, constructing a GP model requires a set of observations $\mathcal{D}$, a mean function $m : \mathbb{R}^d \to \mathbb{R}$ (usually set to zero), and some valid (symmetric and positive semidefinite [30, Ch. 4][2]) kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Using the observed inputs and the kernel function $k$, a Gram matrix [61] $\mathbf{K}$, known as the kernel or covariance matrix, is constructed such that each entry satisfies $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, or

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}. \tag{3.3}$$

Using this kernel matrix and the column vectors composed using the observed outputs

$$\mathbf{y} = \begin{bmatrix} y_1, y_2, \dots, y_n \end{bmatrix}^\top \tag{3.4}$$

and evaluations of the mean function $m(\cdot)$ at the observed inputs

$$\mathbf{m} = \begin{bmatrix} m(\mathbf{x}_1), m(\mathbf{x}_2), \dots, m(\mathbf{x}_n) \end{bmatrix}^\top, \tag{3.5}$$

the equations for the predicted GP mean and variance can be constructed [30, Ch. 2]. Specifically, for a given test point $\mathbf{x}_*$ from Equation 3.2, the equations describing the predicted mean $\mu_{\mathcal{GP}}$ is given by

[2] If this was not the case, some functions may cause the kernel matrix $\mathbf{K}$ to have negative eigenvalues. Since this matrix represents a covariance matrix of a multivariate Gaussian distribution and the eigenvalues of this matrix represent variance along a principal axis, negative eigenvalues would not be readily interpretable.

$$\mu_{\mathcal{GP}}(\mathbf{x}_*) = m(\mathbf{x}_*) + \mathbf{k}_*^\top \mathbf{K}^{-1}(\mathbf{y} - \mathbf{m}) \tag{3.6}$$

and the predicted variance $\sigma^2_{\mathcal{GP}}$ is given by

$$\sigma^2_{\mathcal{GP}}(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{k}_*, \tag{3.7}$$

where the test covariance vector $\mathbf{k}_*$ is defined as

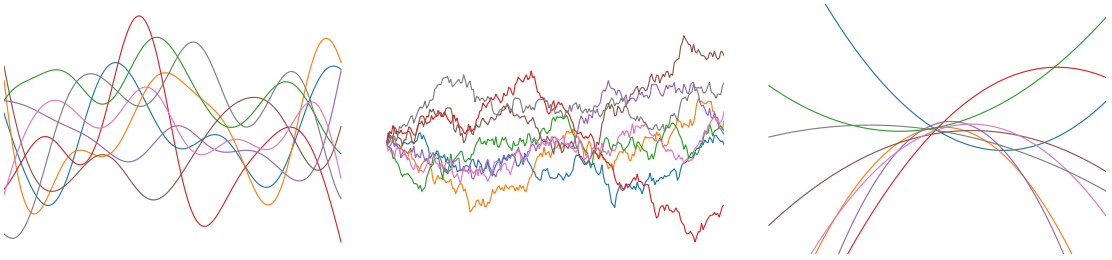
$$\mathbf{k}_* = \left[k(\mathbf{x}_1, \mathbf{x}_*), k(\mathbf{x}_2, \mathbf{x}_*), \ldots, k(\mathbf{x}_n, \mathbf{x}_*)\right]^\top. \tag{3.8}$$

In the calculation of Equations 3.6 and 3.7, determining the inverse matrix $\mathbf{K}^{-1}$ tends to dominate the computation time due to the asymptotic complexity of standard matrix inversion using Gauss-Jordan elimination being $O(n^3)$ for an $n \times n$ matrix [30, Ch. 6]. This is the principal reason why standard BO typically scales poorly with an increasing number of observations $n$. In practical GP implementations, $\mathbf{K}^{-1}$ is rarely used directly. Instead, since $\mathbf{K}$ is known to be symmetric and positive semidefinite due to the use of a valid kernel function [30, Ch. 4], the Cholesky decomposition [62, 63] of the matrix $\mathbf{K}$ is often used in conjunction with forward and backward substitution to indirectly calculate terms containing the inverse. This decomposition is known to be numerically stable and using it in Equations 3.6 and 3.7, while still having a complexity of $O(n^3)$, requires a third of the floating point operations [63] and, therefore, leads to faster wall-clock time prediction performance of the GP.

## 3.2 Kernel Functions

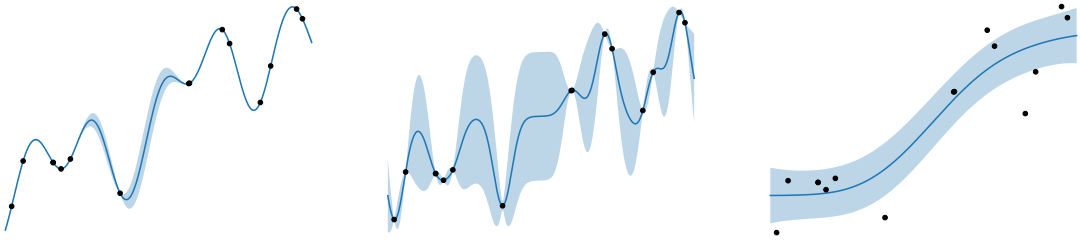
As seen in the previous section, a core component of a GP is the kernel function $k(\cdot, \cdot)$, which quantifies the notion of similarity between points, as it is a reasonable assumption that input points that are similar (in effect, close together) will typically have similar output values. The kernel function is defined as a map from a pair of inputs $\mathbf{x}_p, \mathbf{x}_q \in \mathbb{R}^d$ to $\mathbb{R}$ [64] and must be symmetric positive semidefinite for use in a GP [30, Ch. 4]. The choice of kernel function is also a consequential one, as it defines the family of functions that the GP can represent and, by extension, the shape of the function that will be fitted to the observed points. For example, a GP with a periodic kernel, such as a sinusoidal function, will fit a periodic function to the provided data points. This property is illustrated in Figure 3.2, with samples from the prior distributions for GPs given for three different kernel functions.

Some well-known examples of kernels used in GPs include the Matérn, rational quadratic and squared exponential kernels [30, Ch. 4]. While the Matérn kernel is often the standard choice for global optimization, since it is only twice differentiable versus the infinite differentiability of the squared exponential kernel (a stronger assumption of smoothness that may cause extrapolation issues) [6], the squared exponential kernel may also be a good choice. As the intended use of the GP is to construct local approximations of the objective function, the loss of global interpolative fidelity may be acceptable. The squared exponential kernel also has the useful property that the characteristic *length-scale* parameter $\ell$ of this kernel is strongly correlated with the smoothness of the locally fitted GP model. An example of a GP constructed using a squared exponential kernel with different length-scale values is given in Figure 3.3.



(a) $k(x, x') = \exp\left(\frac{-\|x-x'\|^2}{2\ell^2}\right)$ (b) $k(x, x') = \min(x, x')$ (c) $k(x, x') = (x^\top x' + c)^2$

Figure 3.2: Example showing samples from the GP prior distributions for different kernel functions, namely the (a) squared exponential, (b) Brownian and (c) quadratic kernel functions.



(a) $\ell = 1$ (b) $\ell = 0.3$ (c) $\ell = 3$

Figure 3.3: Example of GPs with the squared exponential kernel for different length-scale values conditioned on the same set of observations. Note that, compared to (a) a baseline length-scale, the shape of the predicted mean becomes (b) more erratic for a shorter length-scale and (c) smoother for a longer length-scale.

Another technique that is often combined with the squared exponential kernel is known as automatic relevance determination, originally proposed by MacKay [56] and Neal [57]. This technique extends the squared exponential kernel with a length-scale parameter for each input dimension $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_d)$, allowing the kernel to model differing amounts of variation and smoothness in each coordinate direction, thereby improving the fidelity of the GP surrogate used by BO to approximate the objective function. The definition of this extended kernel, where $\sigma_f^2$ and $\sigma_n^2$ are the signal and noise variances respectively, is given as

$$k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j) := \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\Lambda}^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right) + \sigma_n^2 \delta_{ij}, \tag{3.9}$$
$$\text{where} \quad \boldsymbol{\Lambda} = \text{diag}(\ell_1^2, \ell_2^2, \ldots, \ell_d^2).$$

In this kernel function, the signal variance $\sigma_f^2$ defines the amplitude of variations for the constructed GP while the noise variance $\sigma_n^2$ defines the measurement noise associated with each input point. The hyperparameters of the squared exponential kernel with automatic

relevance determination $\boldsymbol{\theta}$ are therefore given by

$$\boldsymbol{\theta} = (\sigma_f, \sigma_n, \ell_1, \ell_2, \ldots, \ell_d), \tag{3.10}$$

which fully describe the GP model fit to a set of observed data for the chosen kernel function. As a result, the choice of these parameters directly affect the accuracy of the model in approximating the objective function from the observed data.

It is important to note that while automatic relevance determination enables the modelling of differing smoothness for each *coordinate* direction in the objective function, it cannot do so in *arbitrary* directions. One solution to this problem is the factor analysis distance proposed by Rasmussen and Williams [30, Ch. 5], which adds a product of two low-rank $r \times d$ matrices to the diagonal matrix $\boldsymbol{\Lambda}^{-1}$ in Equation 3.9. While having the advantage of being able to directly infer these arbitrary directions from the observed data, this technique is often prohibitively expensive, requiring an extra $rd$ hyperparameters in addition to the $d$ automatic relevance determination length-scales to model $r$ arbitrary directions. Another solution, presented by Vivarelli and Williams [65], is to replace $\boldsymbol{\Lambda}$ with the product of two triangular matrices (essentially the Cholesky decomposition of $\boldsymbol{\Lambda}$). However, this method is even more expensive, as it requires $\frac{d(d+1)}{2}$ hyperparameters to fill the triangular matrix.

A common choice for the noise variance $\sigma_n$ is a small, fixed value ($\approx 10^{-6}$), known as a "nugget" parameter [37]. This parameter is used to enhance numerical stability, even for deterministic objective functions, as discussed by Gramacy and Lee [37]. To achieve these benefits, the nugget adds a small offset to the diagonal entries of the kernel matrix $\mathbf{K}$, as defined in Equation 3.3, preventing linear dependence between the rows and columns if two input points are nearly equal. This prevents the kernel matrix from becoming singular if the observed points become very correlated. As mentioned in Section 1.1, the addition of this parameter reduces the rate and limit of convergence to a solution, since an artificial level of noise has been superimposed onto the, possibly noiseless, objective function [39].

## 3.3 Hyperparameter Selection

Since the shape of a GP is largely determined by the choice of kernel function $k(\cdot, \cdot)$ and, by extension, the hyperparameters of this kernel function $\boldsymbol{\theta}$, this choice has a significant impact on the quality of the resulting regression model for a set of observed points from an objective function. While this choice of kernel and hyperparameters may be prespecified, Bayesian inference presents an effective approach to determine these values by treating the "best model" (in effect, the hyperparameters of the kernel[3]) as a random variable that must be inferred from the observed data.

Using Bayes' theorem, the posterior distribution of the hyperparameters $\boldsymbol{\theta}$, given the observations $X$ and $Y$, is given by

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(Y \mid X, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid X)}{p(Y \mid X)}, \tag{3.11}$$

[3]Note that while it is possible to perform inference over a set of kernel functions and associated hyperparameters, known as model averaging [27, Ch. 4], this dissertation will focus on inference for a single, chosen kernel function.

where the sets $X = \{\mathbf{x}_i \,|(\mathbf{x}_i, y_i) \in \mathcal{D}\}$ and $Y = \{\mathbf{y}_i \,|(\mathbf{x}_i, y_i) \in \mathcal{D}\}$ are composed of the observed inputs and outputs, respectively.

After making the assumption that the prior distribution over the hyperparameters is independent of the observed inputs ($p(\boldsymbol{\theta} \,|\, X) = p(\boldsymbol{\theta})$) and noting that $p(Y \,|\, X)$ is independent of $\boldsymbol{\theta}$ (in effect, becoming a constant factor), the equation can be simplified as

$$p(\boldsymbol{\theta} \,|\, \mathcal{D}) \propto p(Y \,|\, X, \boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{3.12}$$

where $p(\boldsymbol{\theta})$ is the prior distribution for the hyperparameters that encodes the belief regarding plausible hyperparameters as a probability distribution. This prior may either be informed a priori[4] or be uninformed, in which case it would be specified as $p(\boldsymbol{\theta}) \propto 1$.

By inspecting Equation 3.12 and noting the aforementioned independence of $p(\boldsymbol{\theta})$ of the observed data, it can be concluded that the quality of the model fit is captured by the term $p(Y \,|\, X, \boldsymbol{\theta})$. This term is known as the *marginal likelihood* of the data or *model evidence*. Recalling Equation 3.2, which states that predictions from a GP are described by Gaussian distributions, the marginal likelihood for the observed data can then be described by a multivariate Gaussian distribution $\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$ and, from the logarithmic form of the likelihood function for a multivariate Gaussian, the equation for the log marginal likelihood[5] is given by

$$\log p(Y \,|\, X, \boldsymbol{\theta}) = -\frac{1}{2}(\mathbf{y} - \mathbf{m})^\top \mathbf{K}^{-1}(\mathbf{y} - \mathbf{m}) - \frac{1}{2}\log|\mathbf{K}| - \frac{n}{2}\log 2\pi. \tag{3.13}$$

This expression provides a useful measure to compare different sets of hyperparameters, as the kernel matrix $\mathbf{K}$ is recalculated for different sets of hyperparameters while the rest of the equation remains constant. While the formal Bayesian approach using this measure would be to marginalize over the hyperparameters when making a prediction with the regression model from Equation 3.2, or

$$p(y_* \,|\, \mathbf{x}_*, \mathcal{D}) = \int p(y_* \,|\, \mathbf{x}_*, \mathcal{D}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \,|\, \mathcal{D})\, d\boldsymbol{\theta}, \tag{3.14}$$

this integral is usually intractable [60] and must be integrated numerically using Monte Carlo methods [66, 67] or approximated using the most likely set of hyperparameters[6] $\hat{\boldsymbol{\theta}}$ as

$$p(y_* \,|\, \mathbf{x}_*, \mathcal{D}) \approx p(y_* \,|\, \mathbf{x}_*, \mathcal{D}, \hat{\boldsymbol{\theta}}). \tag{3.15}$$

These hyperparameters $\hat{\boldsymbol{\theta}}$ are often chosen as the maximum of the posterior distribution, known as the maximum a posteriori (MAP) estimate. This MAP estimate is equivalent to maximizing the logarithm of the posterior distribution, or

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}(\log p(Y \,|\, X, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})). \tag{3.16}$$

[4] As noted by Garnett [27], even a very wide prior (which should be possible to construct in practice for any hyperparameters) offers regularization by guiding the model away from patently absurd choices.

[5] The logarithmic form of the likelihood is preferred for computation, since these values can be exceptionally small with a large dynamic range [27, Ch. 4].

[6] This approximation can be interpreted as introducing a Dirac delta function to the marginalization integral Equation 3.14 [27] at the most likely hyperparameters $\delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$ and using the sifting property of this function.

In the case of an uninformative (uniform) prior distribution for the parameters $\boldsymbol{\theta}$, $p(\boldsymbol{\theta})$ would become constant. Therefore, only the log marginal likelihood $\log p(Y \mid X, \boldsymbol{\theta})$ would be maximized, converting this MAP estimate to a maximum likelihood estimate (MLE) [27, Ch. 4].

The optimization problem in Equation 3.16 is usually solved using existing nonlinear solvers (e.g., BFGS [8]), derivative-free methods (e.g., Nelder-Mead [17]) or stochastic methods (e.g., stochastic gradient descent [9]), often with multiple restarts due to the multi-modal nature of the likelihood function. It should be noted that evaluating Equation 3.13 for a new set of hyperparameters also requires recalculating the inverse matrix $\mathbf{K}^{-1}$, which, as previously mentioned, has a computational complexity of the order $O(n^3)$, where $n$ represents the number of observations.

Maximizing the marginal likelihood for GP hyperparameters, therefore, enables their analytical estimation from the observed data for the optimal model fit. By systematically incorporating prior assumptions about the data, this approach not only enhances the GP regression model's ability to generalize but also avoids the pitfalls of manual tuning and extensive kernel engineering.

In summary, this chapter has provided an overview of Gaussian processes (GPs), in particular, expounding on the fundamental components of prediction using a regression model, kernel functions, and hyperparameter estimation and selection. Using these concepts, the Bayesian optimization (BO) framework is constructed and presented in the next chapter. These concepts also underpin the local GP surrogate model used in the proposed LABCAT algorithm presented in Chapter 6–9.

# Chapter 4

# Bayesian Optimization

> GUIL: *And if you'd lost? If they'd come down against you, eighty-five times, one after another, just like that?*
> ROS: *(Dumbly) Eighty-five times in a row? Tails?*
> GUIL: *Yes, what would you think?*
> ROS: *(Doubtfully) Well... (Jocularly) Well, I'd have a good look at your coins for a start!*
>
> — Tom Stoppard, *Rosencrantz and Guildenstern Are Dead*

Having established the structure of sequential model-based optimization (SMBO) methods in Chapter 1 and the probabilistic Gaussian process (GP) surrogate model in Chapter 3, the SMBO method with this GP surrogate, known as Bayesian optimization (BO), can now be described. In this chapter, we review the structure of the core loop of standard BO: the construction of a GP surrogate model described in the previous chapter, finding the next sample point that maximizes an acquisition function, and refining the surrogate model with the sampled point from the objective function. An overview of classical trust-region-based optimization methods and how a trust region is incorporated into the BO loop in the general form of trust-region-based BO algorithms is also provided. This combination of a trust region integrated with BO forms the basis of the proposed LABCAT algorithm presented in Chapter 9 as well as the trust-region-based BO methods previously discussed in Section 2.5.

For further reading regarding the development and use of BO, the reader is directed to the monograph of Garnett [27] and the review conducted by Shahriari et al. [24]. The work of Conn et al. [15] is also recommended for a complete treatment of classical trust-region-based optimization methods.

## 4.1 Standard Bayesian Optimization

Stated formally, the task of black-box optimization algorithms, such as sequential model-based optimization (SMBO) methods in general and BO in particular, can be described as attempting to find some argument $\mathbf{x}_{\min} \in \Omega$ that minimizes a bounded, black-box objective function with continuous input parameters and a scalar output value $f : \mathbb{R}^d \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^d$ is the set of all possible arguments subject to some specified constraints. In this dissertation, it is also assumed that $f$ is observable exactly (in effect, with no noise added to the observed output

values). This optimization task can be stated as calculating

$$\underset{\mathbf{x}_* \in \Omega}{\operatorname{argmin}} f(\mathbf{x}_*), \tag{4.1}$$

with a standard parameterization of $\Omega$ as a hyperrectangle, using a Cartesian product with bounding real, scalar values $\Omega_i^{\min}$ and $\Omega_i^{\max}$ for each dimension, given by

$$\Omega = \prod_{i=1}^{d} [\Omega_i^{\min}, \Omega_i^{\max}], \tag{4.2}$$

with the bounding values subject to

$$\Omega_i^{\min} < \Omega_i^{\max} \quad \forall i \in \{1, \ldots, d\}. \tag{4.3}$$

Since any minimization problem can be converted to a maximization problem with a simple sign change in the objective function, this formulation does not suffer a loss of generality and is a widely-used convention [51–54].

As stated in Chapter 1, BO can be considered as a subset of SMBO methods [22–24] as outlined in Algorithm 1, both using a surrogate model to iteratively approximate the objective function. The distinguishing factor is that the surrogate model used in BO is probabilistic, encapsulating the prior belief regarding the objective function that is sequentially conditioned on new observations of the objective function. While other models have been proposed, such as random forests [22] or Parzen estimators [68], GPs are often the standard choice for the surrogate model. GPs offer proven flexibility, performance and a strong theoretical basis for analysis. They also directly provide both an estimate of the objective function and the uncertainty of this estimate, given by the mean and variance of the GP prediction defined in Equations 3.6 and 3.7.

The second core component of BO is the use of an *acquisition function* (also known as an infill criterion). This function informs the selection of successive input points based on the expected benefit, or *utility*, of a potential evaluation at each input point. Using a generic utility function $U$ defining the gain associated with a single sample according to some metric, the general acquisition function $\alpha$ is defined as

$$\alpha(\mathbf{x}_*; \mathcal{D}) \coloneqq \mathbb{E}_{f(\mathbf{x}_*) \mid \mathbf{x}_*}[U(\mathbf{x}_*, f(\mathbf{x}_*))]. \tag{4.4}$$

Using the probabilistic estimate of the objective function and associated uncertainty thereof from the GP surrogate, the acquisition function can be constructed to formalize the trade-off between exploitation (low predicted GP mean $\mu_{\mathcal{GP}}(\mathbf{x}_*)$) and exploration (high predicted GP variance $\sigma^2_{\mathcal{GP}}(\mathbf{x}_*)$). Maximizing this acquisition function would ideally provide the point that strikes a balance for the aforementioned trade-off when sampled from the objective function and incorporated into the GP surrogate model. This maximization can be stated as

$$\mathbf{x}_\alpha = \underset{\mathbf{x}_* \in \Omega}{\operatorname{argmax}} \alpha(\mathbf{x}_*; \mathcal{D}), \tag{4.5}$$

with the resulting objective function sample $(\mathbf{x}_\alpha, f(\mathbf{x}_\alpha))$ added to the GP surrogate model.

Several utility functions and induced acquisition functions have been investigated for use in BO, such as the entropy search [69] and knowledge gradient [70] acquisition function induced

by the information gain and global simple reward utility functions, respectively. One popular choice of acquisition function is the expected improvement function [71]. This acquisition function is derived from the utility function known as the improvement function that quantifies the amount of improvement on the current best observed candidate solution $(\mathbf{x}_{\min}, y_{\min})$ that has been obtained from an evaluation of the objective function at the given test input. The improvement function is defined as

$$I(\mathbf{x}_*, f(\mathbf{x}_*)) \coloneqq \begin{cases} (y_{\min} - f(\mathbf{x}_*)) & f(\mathbf{x}_*) < y_{\min} \\ 0 & f(\mathbf{x}_*) \geq y_{\min}. \end{cases} \tag{4.6}$$

Recalling from Equation 3.2 that the GP prediction for a test input $\mathbf{x}_*$ is normally distributed, the expected value of the improvement function (in effect, the amount of improvement that is expected at the test input) can be derived analytically as

$$\begin{aligned} \alpha_{\text{EI}}(\mathbf{x}_*; \mathcal{D}) &= \mathbb{E}_{f(\mathbf{x}_*)\,|\,\mathbf{x}_*}[I(\mathbf{x}_*, f(\mathbf{x}_*))] \\ &= (y_{\min} - \mu_{\mathcal{GP}}(\mathbf{x}_*))\,\Phi(z) + \sigma_{\mathcal{GP}}(\mathbf{x}_*)\,\phi(z) \end{aligned} \tag{4.7}$$

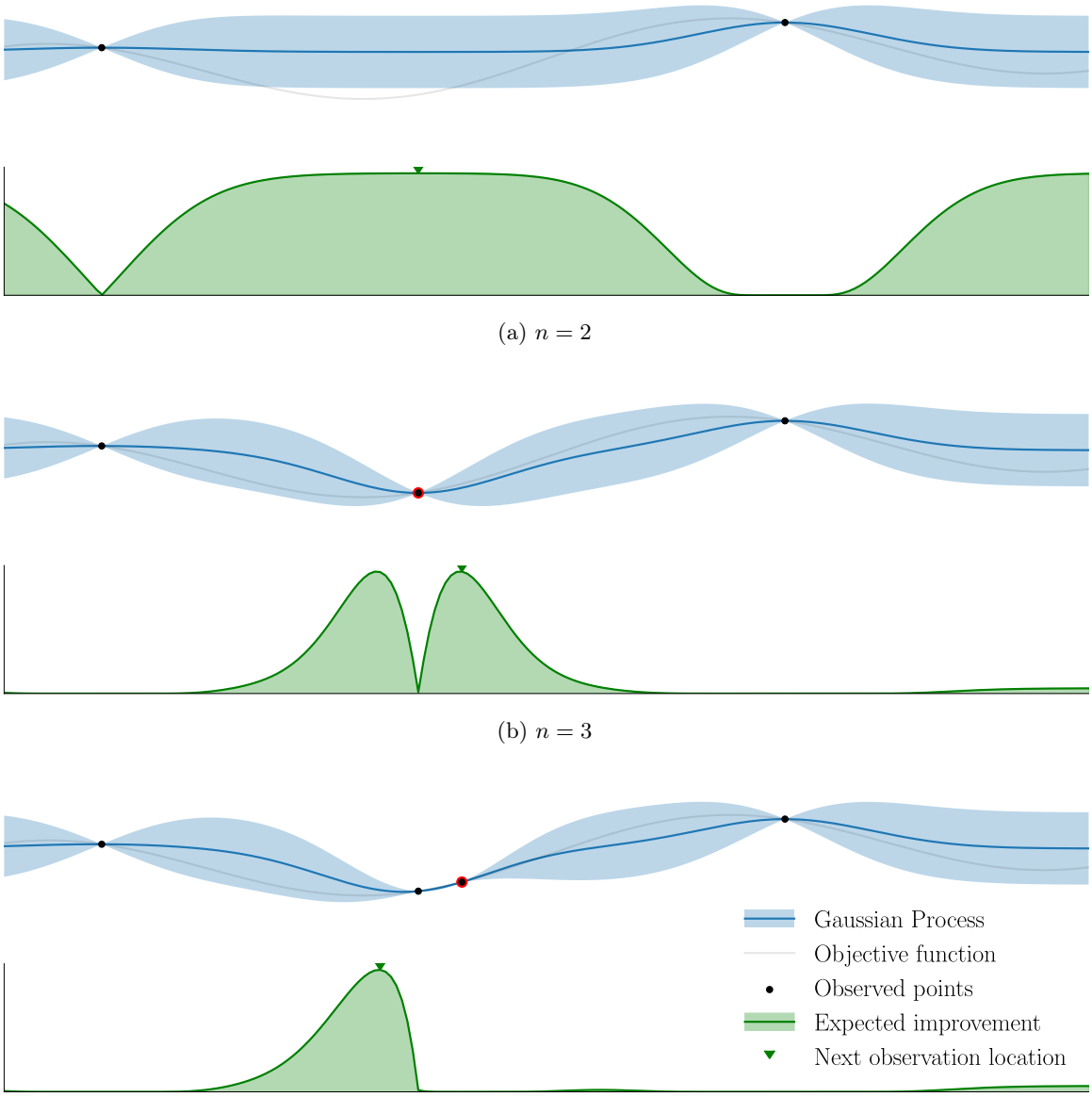
where

$$z = \frac{y_{\min} - \mu_{\mathcal{GP}}(\mathbf{x}_*)}{\sigma_{\mathcal{GP}}(\mathbf{x}_*)}, \tag{4.8}$$

with the probability density function $\phi(z)$ and cumulative distribution function $\Phi(z)$ of a univariate standard normal distribution [71].

Intuitively, this expected improvement acquisition function should be close to zero for any regions where no improvement is expected while the first or second terms would be large where the GP mean is lower than the current minimum (exploitation) or for high GP variances (exploration), respectively. It should be noted that these acquisition functions are often very multimodal and non-convex and, as such, are often maximized using multi-start gradient-based, direct search or evolutionary methods [27, Ch. 9]. Due to the fact that the computationally cheaper GP surrogate model is being sampled by the acquisition function instead of the more expensive objective function, the additional samples often required by these aforementioned methods become computationally viable. A visual example of this process is given in Figure 4.1, with new points successively added to the GP surrogate model by finding the points that maximize the acquisition function.

In the final step of BO, the surrogate model is updated and refined with the new observation. For BO with a GP, this typically entails the reselection of the hyperparameters of the kernel by maximizing the new marginal likelihood of the set of observations using the methods outlined in Section 3.3. As mentioned in Section 1.1, this reselection of the GP hyperparameters does come at the cost of theoretical convergence guarantees [35] compared to using fixed hyperparameters, but the practical performance improvements are often worthwhile. A notable advantage of using fixed hyperparameters is the improved computational complexity, as adding or removing observations to the GP can be efficiently done using the Schur complement [72] (if using $\mathbf{K}^{-1}$ directly) or rank-1 updates and downdates of the Cholesky decomposition of $\mathbf{K}$, both of order $O(n^2)$ [27, Ch. 9]. Reselecting the hyperparameters of the GP would require the full recalculation of the kernel matrix $\mathbf{K}$, precluding the use of these techniques.



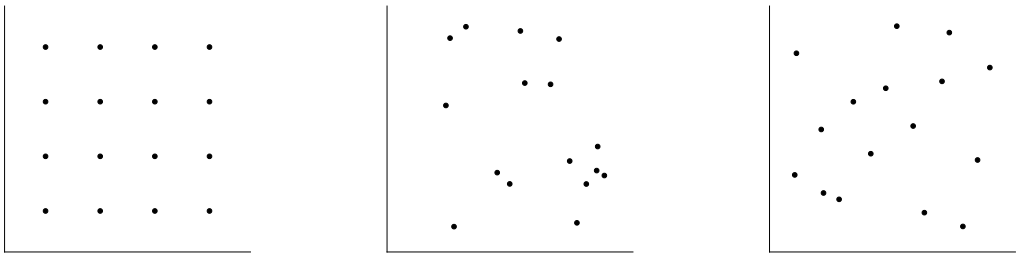

(a) $n = 2$

(b) $n = 3$

(c) $n = 4$

Figure 4.1: Demonstration of several successive iterations of Bayesian optimization (BO) with the expected improvement acquisition function, adapted from [24] and [27]. Note the successive sampled points added to the Gaussian process (GP) model that maximize the acquisition function.

BO is also typically initialized with an initial set of input points before the main loop of BO starts, known as the design of experiment (DoE). A naive approach to selecting these initial points might involve distributing them on a grid, known as full-factorial sampling [38], or randomly. However, both methods have notable shortcomings. Full-factorial sampling, while providing good coverage of the search space $\Omega$, scales exponentially with the number of dimensions, requiring $n^d$ points for $n$ points along each dimension. In contrast, random sampling scales better but lacks coverage guarantees and may result in points clustering together. More sophisticated and better-informed alternative methods, such as Latin hypercube sampling [73] or quasi-random methods like Halton and Sobol sampling [38], are often the typical choice and provide good coverage guarantees while scaling better than full-factorial sampling. Although BO is generally robust to the choice of DoE [23], care must be taken to avoid initial sets of points that may hinder the performance of BO, such as clusters of initial input points far away from the optimum. Examples showing full-factorial, random and Latin hypercube sampling using the same DoE budget is shown in Figure 4.2.



(a) (b) (c)

Figure 4.2: Example of (a) full-factorial sampling, (b) random sampling and (c) Latin hypercube sampling design of experiment (DoE) strategies with identical initial sample budgets.

In general, BO does not have a well-defined, standard stopping criterion, a property shared with other derivative-free optimization methods. This is in contrast to gradient-based methods, where the norm of the gradient can be used to ensure first-order stationarity of the solution. As a result of the lack of a canonical termination criterion, BO is often terminated when a maximum objective function evaluation limit is reached or a minimum decrease condition is satisfied [27, Ch. 9].

Combining the previously discussed initialization of the DoE, construction and refinement of the GP surrogate model, maximization of the acquisition function and termination, the general BO loop is presented in Algorithm 2. The following section expands on this definition of the BO loop with the addition of a trust region, yielding a modified version of this algorithm.

## 4.2 Trust-region-based Bayesian Optimization

As mentioned in Section 2.5, trust-region-based BO methods combine the BO loop described in the previous section with a trust region inspired by those found in classical trust-region-based optimization methods [27, Ch. 11]. These classical methods, also known as restricted-step methods [74], use a linear or quadratic approximation of the objective function that is

**Algorithm 2** Bayesian optimization

**Input:** Objective function $f$, Acquisition function $\alpha$, Bounds $\Omega$, DoE strategy

Select initial input points $X_0$ according to DoE
$\mathcal{D} \leftarrow \{(\mathbf{x}, f(\mathbf{x})) \,|\, \mathbf{x} \in X_0\}$ ▷ Evaluate initial input points from DoE
**while not** convergence criterion satisfied **do**
  $\mathcal{GP}(m(\cdot), k(\cdot,\cdot); \mathcal{D})$ ▷ Construct GP with observed data $\mathcal{D}$
  $\hat{\boldsymbol{\theta}} \leftarrow \operatorname{argmax}_{\boldsymbol{\theta}}(\log p(Y \,|\, X, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}))$ ▷ Determine GP hyperparameters
  $\mathbf{x}_\alpha \leftarrow \operatorname{argmax}_{\mathbf{x}_* \in \Omega} \alpha(\mathbf{x}_*; \mathcal{D})$ ▷ Maximize acquisition function
  $y_\alpha \leftarrow f(\mathbf{x}_\alpha)$ ▷ Evaluate suggested input point
  $\mathcal{D} \leftarrow \{(\mathbf{x}_\alpha, y_\alpha)\} \cup \mathcal{D}$ ▷ Add observation to dataset
**end while**
**return** $(\mathbf{x}_{\min}, y_{\min})$ ▷ Return minimum candidate

"trusted" for a local region surrounding the current best candidate solution [15], hence the term *trust region* coined by Sorensen [75]. New points to sample are selected by finding the optimum of this approximation of the objective function constrained by the trust region, with the trust region (and by extension, the approximation) being expanded or contracted as the algorithm executes, allowing for different step sizes between subsequent candidate solutions. In some sense, these methods are the complement of line search methods (such as gradient descent), first determining the step size and then the step direction, instead of the reverse in line search.

While the trust regions used by these classical methods can theoretically be any shape [15, Ch. 6], they are often spherical (defined by a radius) or hypercubic (with a side length)[1] that can be extended with independent scaling for each axis to ellipsoidal or rectangular regions, respectively. While spherical and ellipsoidal trust regions are often preferred for classical trust-region-based algorithms due to theoretical advantages [15, Ch. 7],[2] square and rectangular trust regions are often easier to use, as points can simply be checked axis by axis. The trust region may also be rotated, often using the Hessian of the objective function or approximations thereof, to allow the trust region to better adapt to the objective function [15, Ch. 6].

Generally, the distinguishing factor between different trust-region-based methods are the strategies used to update the size of the trust region. These may be a simple improvement heuristic that increases or decreases the size of the trust region at the current algorithm iteration $s_t$ by constant factors $\kappa$ and $\eta$ based on if the next sampled point is better than the current minimum candidate

$$s_t \mapsto s_{t+1} = \begin{cases} \frac{1}{\kappa} s_t & f(\mathbf{x}_{\min}) > f(\mathbf{x}_\alpha) \\ \eta s_t & f(\mathbf{x}_{\min}) < f(\mathbf{x}_\alpha). \end{cases} \tag{4.9}$$

Other heuristics may be based on how well the trust region is approximating the true objective function [15, Ch. 6], such as the heuristic in the Levenberg-Marquardt algorithm [13]. These

[1] These trust region shapes naturally arise from the $L^2$ or $L^1$ and $L^\infty$ norms, respectively [15, Ch. 6.1]. Ellipsoidal and rectangular trust regions can also be expressed using scaled versions of these norms [15, Ch. 6.7].

[2] Specifically, finding an optimum point using a spherical/ellipsoidal trust region with linear or quadratic approximations requires at most polynomial time, while using a square/rectangular trust region may be an NP-hard problem [15, Ch. 7.8].

heuristics check if the ratio $\varrho = \frac{\Delta f_{\text{pred}}}{\Delta f_{\text{actual}}}$ of the difference between the value of the current minimum candidate solution and the value of the next point predicted by the trust region approximation $\Delta f_{\text{pred}}$ and the actual, measured difference $\Delta f_{\text{actual}}$ is above or below a certain threshold

$$s_t \mapsto s_{t+1} = \begin{cases} \frac{1}{\kappa} s_t & \varrho > \varrho_2 \\ s_t & \varrho \in [\varrho_1, \varrho_2] \\ \eta s_t & \varrho < \varrho_1, \end{cases} \tag{4.10}$$

with typical values of $\varrho_1 = 0.25, \varrho_2 = 0.75, \kappa = 3$ and $\eta = 2$ [76]. While the factors $\kappa$ and $\eta$ are typically constant, some methods allow these factors to be dynamically changed using an additional heuristic such as dynamic values based on $\varrho$ or the previous step size [15, Ch. 10].

As noted at the beginning of this section, trust-region-based BO methods (examples of which were previously discussed in Section 2.5) synthesize the idea of classical trust-region-based methods with BO and can be seen as either substituting the linear or quadratic objective function approximations of classical trust-region-based methods with a Gaussian process surrogate model or bounding the acquisition function maximization step of BO using a trust region [27, Ch. 11]. This combination can be formalized [51–53] by adding an additional constraint to the maximization of the acquisition function from Equation 4.5, bounding the feasible region of this maximization using the intersection $\Omega \cap \Omega_{\text{TR}}$ of the trust region $\Omega_{\text{TR}}$ and the original bounds of the objective function $\Omega$. Including this additional constraint in Equation 4.5 is given as

$$\mathbf{x}_\alpha = \underset{\substack{\mathbf{x}_* \in \Omega_{\text{TR}} \\ \mathbf{x}_* \in \Omega}}{\operatorname{argmax}} \alpha(\mathbf{x}_*; \mathcal{D}). \tag{4.11}$$

Incorporating this trust region modification into the standard BO algorithm from Algorithm 2 yields a general trust-region-based BO loop in Algorithm 3 with examples of these methods previously discussed in Section 2.5. The primary addition in this form of trust-region-based BO compared to the standard BO loop can be seen in the trust region $\Omega_{\text{TR}}$ that is initialized at the start of the algorithm (line 3), used as an additional constraint for the acquisition function maximization (line 7), and updated at the end of each iteration (line 10).

Trust-region-based BO methods often use the information encoded in the BO loop to derive trust region update strategies, such as the ratio of expected versus actual improvement [52], which is similar to the ratio $\varrho$ used in Equation 4.10. It should also be noted that, in the trust-region-based BO context, the shape of the trust region does not matter as much as for classical trust-region-based methods, since the advantages of spherical trust regions (as stated in Footnote 2) no longer hold, given that maximizing the acquisition function is often an NP-hard problem [77]. Therefore, square and rectangular trust regions are the standard choices [50–55], as these trust regions are easier to work with.

As previously stated in Sections 2.5 and 2.6, trust-region-based BO methods have proven to be more resilient to badly-behaved objective functions than standard BO. Compared to classical trust-region-based optimization methods, the use of a more informed GP surrogate model, instead of a simple linear or quadratic model, also allows for improved robustness and sample efficiency [55]. While the trust region approach means that trust-region-based BO methods relax the global perspective of standard BO (with the associated global optimization

**Algorithm 3** Trust-region-based Bayesian optimization

**Input:** Objective function $f$, Acquisition function $\alpha$, Bounds $\Omega$, DoE strategy

Select initial input points $X_0$ according to DoE
$\mathcal{D} \leftarrow \{(\mathbf{x}, f(\mathbf{x})) \,|\, \mathbf{x} \in X_0\}$ ▷ Evaluate initial input points from DoE
Initialize trust region $\Omega_{\text{TR}}$
**while not** convergence criterion satisfied **do**
    $\mathcal{GP}(m(\cdot), k(\cdot,\cdot); \mathcal{D})$ ▷ Construct GP with $X$ and $Y$
    $\hat{\boldsymbol{\theta}} \leftarrow \text{argmax}_{\boldsymbol{\theta}}(\log p(Y \,|\, X, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}))$ ▷ Determine GP hyperparameters
    $\mathbf{x}_\alpha \leftarrow \text{argmax}_{\substack{\mathbf{x}_* \in \Omega_{\text{TR}} \\ \mathbf{x}_* \in \Omega}} \alpha(\mathbf{x}_*; \mathcal{D})$ ▷ Maximize bounded acq. function
    $y_\alpha \leftarrow f(\mathbf{x}_\alpha)$ ▷ Evaluate suggested input point
    $\mathcal{D} \leftarrow \{(\mathbf{x}_\alpha, y_\alpha)\} \cup \mathcal{D}$ ▷ Add observation to dataset
    Update trust region size using $(\mathbf{x}_\alpha, y_\alpha)$
**end while**
**return** $(\mathbf{x}_{\min}, y_{\min})$ ▷ Return minimum candidate

performance) to a series of robust[3] local approximations [51], this is often sufficient for many problems in practice. Existing trust-region-based BO methods also lack a similar mechanism to rotate the trust region, as used in several classical trust-region-based methods [13], due to the lack of gradient and Hessian information of black-box objective functions.

Taken as a whole, the ideas presented in this chapter, namely the Bayesian optimization (BO) framework in general and the trust-region-based BO subclass in particular, form the cornerstone of the proposed LABCAT algorithm and the context in which this dissertation is written. These BO methods are powerful techniques for efficiently optimizing black-box functions by balancing exploration and exploitation through probabilistic surrogate models, with trust-region-based methods extending this framework with an adaptive restriction of the exploration to regions where improvement is likely. These methods form the basis for the contributions in Chapter 6–9 as well the methods previously discussed in Section 2.5 of the literature study of Chapter 2 and used in the comparative study in Chapter 10.

[3]Robust in the sense that the local GP surrogate model is more resilient to the model misspecification encountered by classical trust-region-based methods as well as the capacity of the GP for the heterogeneous modelling of non-stationary objective functions that can allow the method to avoid patently suboptimal local minima.

# Chapter 5

# Principal Components

Several classical trust-region-based optimization methods, such as Levenberg-Marquardt [13], incorporate rotation of the trust region to allow for better adaptation of the objective function approximation model to the local geometry of the objective function [15, Ch. 6]. For example, this may include adapting to separability found along non-coordinate axes, such as diagonal valleys in the objective function. These trust region rotations are often based on the Hessian of the objective function or an approximation thereof using the Jacobian, finite differences or secant methods. Of course, in the context of black-box optimization problems, information regarding the Jacobian and Hessian is not directly available and approximations thereof may be computationally expensive or intractable to calculate.

To obtain the advantages associated with a rotatable trust region, a rotation at each iteration of the LABCAT algorithm is included using weighted principal components such that the observed data is decorrelated along the axes of the trust region (Chapter 7). Therefore, this chapter reviews the derivation of principal components and the use thereof for a decorrelation transform in Section 5.1, as well as the extension of principal components using per-observation or per-dimensional weights in Section 5.2.

## 5.1 Standard Principal Components

Principal components are a set of directions derived from principal component analysis (PCA), a well-studied statistical technique often used for dimensionality reduction with the earliest generally accepted descriptions presented by Pearson [78] and Hotelling [79]. For a more complete and authoritative taxonomy of PCA, readers are directed to the work of Jolliffe [80].

In essence, PCA identifies the orthogonal directions (in effect, the principal components) along which the variance of a set of sampled data is maximized. The first principal component captures the most variance, followed by the second principal component, which is also orthogonal to the first, and so on. These principal components often capture the most important features of the data and can be thought of as "important" or "essential" directions in the dataset, which would consist of the observed inputs of the objective function in the optimization context of this dissertation.

The standard derivation of PCA, as presented by Hotelling [79], consists of determining the principal components sequentially. The principal components are determined by projecting the sampled points $\mathbf{x}_i \in \mathbb{R}^d \ \forall i \in \{1, \ldots, n\}$, which are collected into the columns of the data

matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, onto the vector $\mathbf{u} \in \mathbb{R}^{d \times 1}$, given by[1]

$$\begin{aligned} \text{var}(\mathbf{u}^\top \mathbf{X}) &= \text{cov}(\mathbf{u}^\top \mathbf{X}, \mathbf{u}^\top \mathbf{X}) \\ &= \mathbf{u}^\top \text{cov}(\mathbf{X}, \mathbf{X}) \mathbf{u} \\ &= \mathbf{u}^\top \mathbf{C}_{\mathbf{XX}} \mathbf{u}, \end{aligned} \tag{5.1}$$

where the sample covariance matrix $\mathbf{C}_{\mathbf{XX}} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ when $\mathbf{X}$ is assumed to be zero-mean. The first principal component $\mathbf{u}_1$ is determined by maximizing this projected variance, where $\mathbf{u}_1$ is constrained to be unit length, or

$$\begin{aligned} \mathbf{X}\mathbf{u}_1 = \underset{\mathbf{u}}{\text{argmax}}\, & \mathbf{u}^\top \mathbf{C}_{\mathbf{XX}} \mathbf{u} \\ \text{subject to } & \|\mathbf{u}_1\| = 1. \end{aligned} \tag{5.2}$$

This constrained optimization problem can be solved by incorporating the constraint using a Lagrange multiplier [12, Ch. 5] into a Lagrangian:

$$L(\mathbf{u}_1, \lambda) = \mathbf{u}_1^\top \mathbf{C}_{\mathbf{XX}} \mathbf{u}_1 - \lambda(\mathbf{u}_1^\top \mathbf{u}_1 - 1). \tag{5.3}$$

By setting the partial derivative of this Lagrangian function to zero

$$\frac{\partial L}{\partial \mathbf{u}_1} = 2\mathbf{C}_{\mathbf{XX}} \mathbf{u}_1 - 2\lambda \mathbf{u}_1 = 0 \tag{5.4}$$

and rearranging terms, we obtain

$$\mathbf{C}_{\mathbf{XX}} \mathbf{u}_1 = \lambda \mathbf{u}_1, \tag{5.5}$$

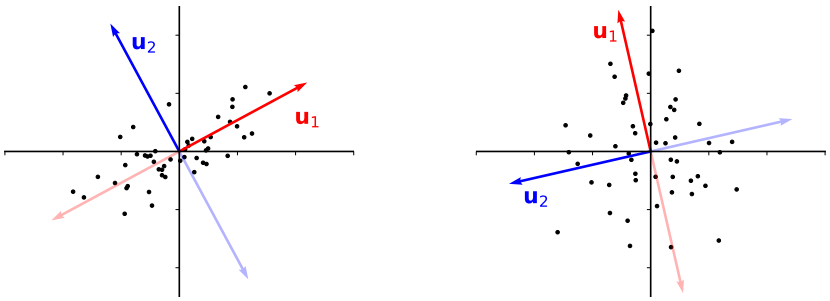
showing that $\mathbf{u}_1$ is an eigenvector of $\mathbf{C}_{\mathbf{XX}}$ with a corresponding eigenvalue of $\lambda_1$. A similar process is used to determine successive principal components by adding orthogonality constraints to the Lagrangian defined in Equation 5.3, for example, adding the constraint $\mathbf{u}_2^\top \mathbf{u}_1 = 0$ when calculating $\mathbf{u}_2$.

This process is equivalent to determining the eigendecomposition (also known as the spectral decomposition) of the sample covariance matrix

$$\mathbf{C}_{\mathbf{XX}} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top, \tag{5.6}$$

where the eigenvectors (columns of $\mathbf{Q}$) are ordered according to their respective eigenvalues (diagonal values of $\mathbf{\Lambda}$ with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$), an indication of the total variance that each eigenvector accounts for. This ordered, orthogonal matrix $\mathbf{Q}$ now defines the principal components of the data matrix $\mathbf{X}$ with the respective diagonal values of $\mathbf{\Lambda}$ being the relative importance of each principal component. An example showing the principal components for two sets of simulated data in shown in Figure 5.1.

[1] Note that the covariance is a linear operator, implying the identity $\text{cov}(\mathbf{a}^\top \mathbf{X}, \mathbf{b}^\top \mathbf{X}) = \mathbf{a}^\top \text{cov}(\mathbf{X}, \mathbf{X})\mathbf{b}$.




(a) (b)

Figure 5.1: Example showing the principal components for two sets of simulated data points. It is clear that the first principal component $\mathbf{u}_1$ captures more projected variance (in effect, the dominant direction of the data points) than the second principal component $\mathbf{u}_2$ in (a) compared to (b), where the choice of principal components seem more ambiguous due to the lack of dominant directions in the data points.

A computationally efficient alternative to the eigendecomposition can be found in the singular value decomposition (SVD) [81], which can be seen as an extension of the eigendecomposition to non-square matrices. Compared to the eigendecomposition, this decomposition has increased numerical stability and is more computationally efficient without requiring the calculation of $\mathbf{C_{XX}}$. The SVD of the data matrix $\mathbf{X}$ is defined as

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \tag{5.7}$$

where $\mathbf{U} \in \mathbb{R}^{d\times d}$, $\mathbf{V} \in \mathbb{R}^{n\times n}$ and $\mathbf{\Sigma}$ is a rectangular diagonal matrix of singular values, with the columns of these matrices also ordered according to the singular values similarly to the ordering of Equation 5.6.[2] In this decomposition, the principal components are given by the columns of $\mathbf{U}$ and are equivalent to the corresponding columns of matrix $\mathbf{Q}$ (up to a sign change) in the eigendecomposition from Equation 5.6. The square of the singular values are also proportional to the eigenvalues and similarly indicates the proportion of variance in the dataset that each principal component accounts for.

From another, equivalent perspective, the principal components can also be used to define an optimal low-rank approximation $\tilde{\mathbf{X}}$ of the original data matrix $\mathbf{X}$ that can be calculated using a reduced SVD

$$\tilde{\mathbf{X}} = \mathbf{U}_{[r]}\mathbf{\Sigma}_{[r]}\mathbf{V}_{[r]}^\top, \tag{5.8}$$

where $[r]$ indicates the upper $r \times r$ block of $\mathbf{\Sigma}$ and the first $r$ columns of $\mathbf{U}$ and $\mathbf{V}$, respectively. This approximation can be proven to minimize the reconstruction error [80], in this case the total element-wise squared difference between the matrices, parameterized by the squared Frobenius norm and is given by

$$\left\|\mathbf{X} - \tilde{\mathbf{X}}\right\|_F^2 = \sum_{i=1}^{d}\sum_{j=1}^{n}(X_{ij} - \tilde{X}_{ij})^2, \tag{5.9}$$

[2]Note that if $\mathbf{X}$ is a matrix with real values, $\mathbf{U}$ and $\mathbf{V}$ are also guaranteed to be real, orthogonal matrices.

for all matrices in $\mathbb{R}^{d\times n}$ of rank $r$. In other words, $\tilde{\mathbf{X}}$ is a least-squares optimal representation of the original $d$-dimensional data in an $r$-dimensional subspace with monotonically decreasing reconstruction error for an increase of $r$ up to $d$.

If all principal components are retained (or, $r = d$), instead of dimensionality reduction, a change of basis is performed such that the data is decorrelated. This is used in a process known as *whitening*, where the data is first decorrelated and then rescaled such that the covariance matrix is the identity matrix $\mathbf{I}$. Using the eigendecomposition defined in Equation 5.6 after calculating the covariance matrix $\mathbf{C}_{\mathbf{XX}}$, this whitening transform is defined as[3]

$$\mathbf{X} \mapsto \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^\top\mathbf{X} \tag{5.10}$$

or, using the SVD from Equation 5.7 directly,

$$\mathbf{X} \mapsto \frac{1}{\sqrt{n}}\mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{X}. \tag{5.11}$$

These whitening transforms can readily be interpreted as a linear transformation with rotational (orthogonal $\mathbf{Q}^\top$ and $\mathbf{U}^\top$ matrices) and scaling components (diagonal $\mathbf{\Lambda}^{-\frac{1}{2}}$ and $\mathbf{\Sigma}^{-1}$ matrices). As seen in the example in Figure 5.2, the rotational component rotates the data to align the principal components with the coordinate axes and the scaling component rescales the data to unit variance.

## 5.2 Weighted Principal Components

During the calculation of the standard principal components defined in the previous section, all of the samples (columns of the data matrix $\mathbf{X}$) are treated equally for the purposes of determining the principal components. However, this uniform treatment may be restrictive and is not always desirable. In the optimization context of this dissertation, where each sample represents an observed input of the objective function with an associated output value, it is preferable to account for the relative importance of each observation. Incorporating this importance into the calculation of principal components would enhance the ability of the principal components to capture the key, underlying directions of the objective function, thereby identifying more effective search directions during the optimization process.

To address the limitations of treating all samples equally in PCA, weighted principal component analysis (weighted PCA) is commonly employed. In weighted PCA, each sample or dimension is assigned a weight that reflects its relative importance or relevance within the dataset. This approach allows for a more nuanced calculation of *weighted* principal components, where the chosen weights influence the determination of these components. By incorporating these weights, weighted PCA modifies the standard principal component calculations to better capture the directions that are most significant according to the assigned weights.

We will first consider the generalized decomposition defined by Greenacre [82, App. A], which introduces the positive-definite, symmetric weight matrices $\mathbf{\Psi}$ and $\mathbf{\Phi}$ that are multiplied with the data $\mathbf{X}$. Using the SVD, this general definition is given as

[3] In these transforms, anonymous function notation is used to reduce symbolic overhead. Read as 'maps to', the use of this arrow notation avoids the need to name these functions. For example, Equation 5.10 would otherwise need to introduce some new symbol, such as the function $g$, to write $g(\mathbf{X}) = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^\top\mathbf{X}$.

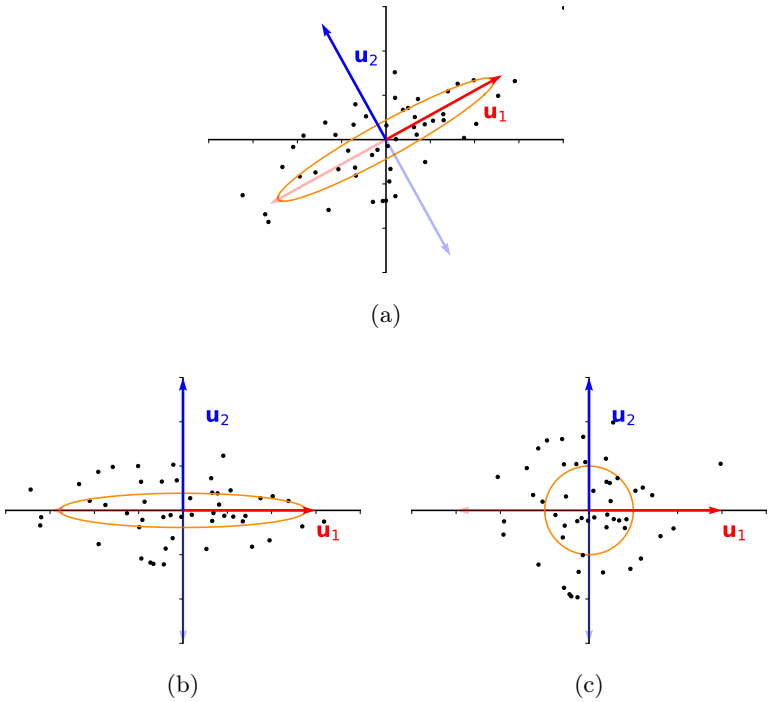



Figure 5.2: Example showing the whitening transform using principal components applied to a set of simulated data. The principal components of the simulated data $\mathbf{u}_1$ and $\mathbf{u}_2$ in (a) are first rotated to align with the coordinate axes in (b) before being rescaled to unit variance in (c). The covariance of the data is also displayed using an ellipse rescaled to a sphere.

$$\mathbf{\Psi}^{\frac{1}{2}}\mathbf{X}\mathbf{\Phi}^{\frac{1}{2}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}, \tag{5.12}$$

which, after rearranging terms, can also be given by

$$\mathbf{X} = \mathbf{\Psi}^{-\frac{1}{2}}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}\mathbf{\Phi}^{-\frac{1}{2}}. \tag{5.13}$$

In the special case where $\mathbf{\Psi}$ and $\mathbf{\Phi}$ are diagonal matrices with positive diagonal elements $\{\psi_1 \dots \psi_d\}$ and $\{\phi_1 \dots \phi_n\}$, respectively, it is also shown by Greenacre that performing a similar low-rank approximation to Equation 5.8

$$\tilde{\mathbf{X}}_w = \mathbf{\Psi}^{-\frac{1}{2}}\mathbf{U}_{[r]}\mathbf{\Sigma}_{[r]}\mathbf{V}_{[r]}^{\top}\mathbf{\Phi}^{-\frac{1}{2}} \tag{5.14}$$

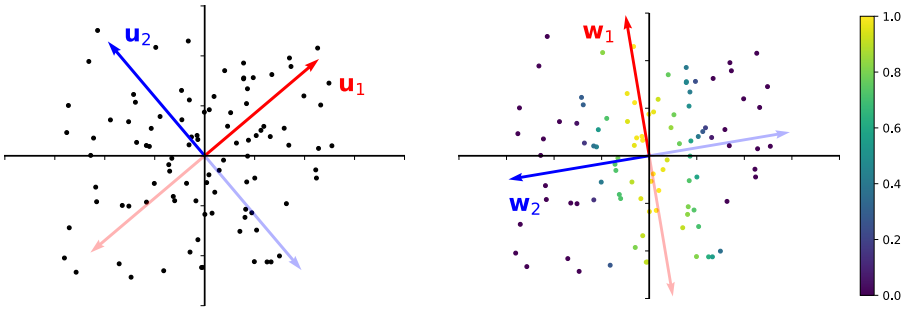
minimizes

$$\left\|\mathbf{X} - \tilde{\mathbf{X}}_w\right\|_F^2 = \sum_{i=1}^{d}\sum_{j=1}^{n}\psi_i\phi_j(X_{ij} - \tilde{X}_{ij})^2, \tag{5.15}$$

providing the best rank $r$ approximation to $\mathbf{X}$ subject to $\mathbf{\Psi}$ and $\mathbf{\Phi}$, in the least squares sense. Compared to the similar standard form of PCA from Equation 5.9, it is clear this special case

of generalized PCA is a form of weighted PCA, where $\psi_i$ assigns weights to each dimension of the observations and $\phi_j$ assigns weights to each observation.

Using this formulation of weighted PCA, the weight matrices can be set to readily account for missing observations [83], repeated observations [84], relative observation population sizes [85], measurement noise [83] or outliers [84]. An example of weighted PCA on a set of simulated observations is given in Figure 5.3.




(a) (b)

Figure 5.3: Example showing (a) the standard principal components for a set of data points ($\mathbf{u}_1$ and $\mathbf{u}_2$) and (b) the weighted principal components for the same set ($\mathbf{w}_1$ and $\mathbf{w}_2$) using additional sample weight information (in this example, the observed output values of some objective function). This example illustrates that, using additional weight information, different principal components can be obtained from the same data points and that the weighted principal components can describe the underlying structure of a set of observed data, in this case, a diagonal ridge.

While several general choices of $\mathbf{\Psi}$ and $\mathbf{\Phi}$ have been suggested, such as the linear, exponential or Gaussian functions used by Krigel et al. [84], Jolliffe [80] posits that 'in practice, it must be rare that an obvious uniquely appropriate set of the $\psi_i$ or $\phi_j$ is available, though a general pattern may suggest itself'. This behaviour was also observed by Krigel et al. [84], who noted that 'it is plausible that different functions are appropriate for different underlying causes in the data or assumptions'. While Hong et al. [86] proved that the optimal choice of weights converges to a function of known or estimated signal and noise variances for high-dimensional data under certain statistical assumptions, these assumptions cannot be made in the context of black-box optimization for this dissertation. Therefore, the lack of a generally optimal choice of these weights suggests that any method to determine the values of these weights should generally be founded on sensible assumptions and supported empirically.

It should also be noted that the intersection of Bayesian optimization (BO) and PCA is not completely uncharted. However, rather than the novel trust region rotation using weighted principal components performed by the LABCAT algorithm, existing research is focused on high-dimensional BO using low-rank principal-component-based linear embeddings [87] or non-linear embeddings [88] to identify low-dimensional manifolds with which to perform BO inside the high-dimensional objective function in order to reduce the effective dimensionality of the search space.

In summary, this chapter presents the concept of principal components (PCs) as generally used in PCA, as well as the special case of generalized PCA with observational and dimensional weights. The principal components from this special case of weighted PCA are used in the proposed trust-region rotation scheme outlined in Chapter 7, with the choice of these weights also verified empirically using numerical benchmarks in Chapter 10.

# Chapter 6

# Overview of the LABCAT Algorithm

The previous chapters have now sufficiently discussed the context, background and prerequisite knowledge required to mark the delineation between the prior art and novel contributions presented in this dissertation. Building on the foundations established in the preceding chapters, this chapter provides a high-level overview of the structure and motivation of the novel black-box optimization method proposed in this dissertation (with detailed mathematical descriptions thereof reserved for the subsequent chapters), which is denominated as the *locally adaptive Bayesian optimization using principal-component-aligned trust regions* (LABCAT) algorithm. A flowchart describing this algorithm is given in Figure 6.1, with the novel additions to the general trust-region-based Bayesian optimization (BO) algorithm from Algorithm 3 given by the shaded components.

When inspecting Figure 6.1, it can be seen that the LABCAT algorithm enters a modified version of the standard trust-region-based BO loop after evaluating the initial set of input points, with the main modifications consisting of transforming the observed data and discarding observations that fall outside the trust region. At the start of the modified trust-region-based BO loop, the current set of observed inputs are recentred on the current minimum candidate and rotated to align the principal components of the observed input points (weighted by the corresponding normalized observed output values) with the coordinate axes. Using this recentred, rotated and normalized observed input and output data, a GP with a squared exponential kernel and automatic relevance determination is constructed. The MAP estimate for the length-scales of this kernel is used to rescale the observed input data, updating the size of the current trust region. To prevent the unbounded growth in complexity of the GP model, an approximate model fitted to the local subset of observations inside the current trust region is maintained by discarding observations outside of the trust region. Finally, the expected improvement acquisition function is maximized, bounded by the trust region, to determine the next observed point.

The LABCAT algorithm follows the example of other trust-region-based Bayesian optimization (BO) algorithms, described in Algorithm 3 of Section 4.2, by incorporating a local trust region surrounding the current minimum candidate solution to bound the acquisition function maximization during the determination of subsequent input points in the standard BO loop [50–55]. What distinguishes LABCAT is that the size of this local trust region is selected to be directly proportional to the local length-scales of the Gaussian process (GP) fitted to the observed data instead of according to a progress-based or sufficient decrease heuristic.

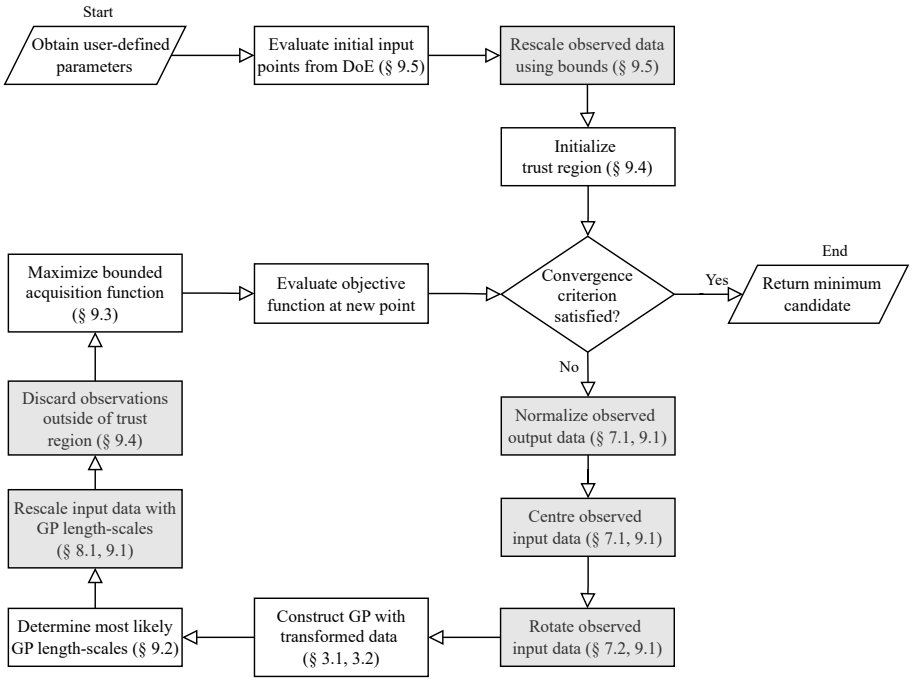



Figure 6.1: A flowchart of the LABCAT algorithm. The primary added components of the LABCAT algorithm, compared to the standard trust-region-based BO described in Algorithm 3, are indicated by the shaded areas and the relevant sections for each component are indicated in brackets. A complete mathematical description of the LABCAT algorithm is given in Chapter 9 along with Algorithm 4.

Furthermore, the trust region is also rotated to align with the weighted principal components of the observed data, allowing the side lengths of the trust region to change independently along arbitrary directions, not just along the coordinate axes. This novel rotation and rescaling also constitute another distinguishing characteristic of the LABCAT algorithm: In the proposed algorithm, the size of the trust region remains *fixed* while the space of observed points is re-centred, rotated and rescaled, indirectly inducing a dynamic trust region in the original space of the objective function. This stands in contrast to other trust-region-based methods that manipulate the size of the trust region directly. Finally, observations that fall outside of the local trust region as new minimum candidates are found are also greedily discarded, preventing the unbounded growth of the surrogate model and improving the convergence characteristics of the LABCAT algorithm. This observation discarding procedure is also in contrast to the previously noted methods that either retain all points [50–53, 55] or employ a significant subset of the evaluation history [54] in the surrogate model.

The following two chapters extend the trust-region-based BO framework described in Algorithm 3 by incorporating a rotation based on the weighted principal components (Chapter 7) and a length-scale-based rescaling (Chapter 8) of the observed data. Together, these extensions culminate in the proposed LABCAT algorithm, the detailed mathematical description of which is given in the subsequent chapter (Chapter 9).

# Chapter 7

# Weighted-principal-component-based Rotation

As noted in Section 4.2, several trust-region-based Bayesian optimization (BO) methods use automatic relevance determination to allow the side lengths of the trust region to be changed independently for each dimension [50, 51, 55]. This mechanism allows the trust region to expand and contract in directions for which the objective function may be smoother and vice versa. The independent expansion and contraction of the trust region side lengths makes these algorithms well suited to separable objective functions that can be broken down into a series of subproblems along independent, orthogonal directions such as ellipsoidal functions, with the main shortcoming of this approach being that the trust region resizing in these methods is limited to the directions defined by the coordinate axes. Ideally, the trust region should be allowed to rotate and dynamically align itself with directions of separability in the objective function, for example, along changing valleys. This expansion of the trust region along the directions of separability would, in turn, allow for larger step sizes between subsequent sampled points and lead to faster convergence towards the optimum. As mentioned in Chapter 4, while several classical trust-region-based optimization methods incorporate a rotation of the trust region using the Hessian of the objective function (or an estimate thereof), this information is not available or intractable to estimate for black-box objective functions.

Instead of rotating the trust region directly, this chapter introduces a preprocessing step that involves a rotation based on the weighted principal components of the observed data weighted according to their objective function values. By first aligning the observed data with these weighted principal components and then using an axis-aligned trust region in conjunction with the transformed data, this achieves the benefits of a rotatable trust region while being computationally more efficient than methods that manipulate the trust region directly, such as using the costly factor analysis distance [30, Ch. 5] discussed in Section 3.2.

This chapter first describes the general technique of data preprocessing as a transformation with an associated invariant property. It then covers the transformations and invariant properties of two standard preprocessing techniques, known as min-max normalization and centring respectively, as well as the proposed rotation. Finally, it elucidates the advantage of this rotation within a trust-region-based BO framework through an illustrative example.

## 7.1 Data Preprocessing In Trust-region-based Bayesian Optimization

Raw data is often transformed into a form that is more suitable for analysis or manipulation, typically involving scaling, normalizing, or rotating the data. This step, known as data preprocessing [89] or preconditioning, helps improve the accuracy and efficiency of models constructed using the transformed data by addressing issues such as ill-conditioning, numerical issues, or non-standard orientations. By incorporating a preconditioning step at the start of the algorithm or before each iteration of the trust-region-based BO framework from Algorithm 3, illustrated in Figure 7.1, the performance of the algorithm may benefit from the increased quality of the surrogate model constructed using the preprocessed, observed data. A popular use of these techniques in trust-region-based BO methods is to scale the observed data such that the bounds of the objective function $\Omega$ lie on a unit hypercube [51, 52, 54].

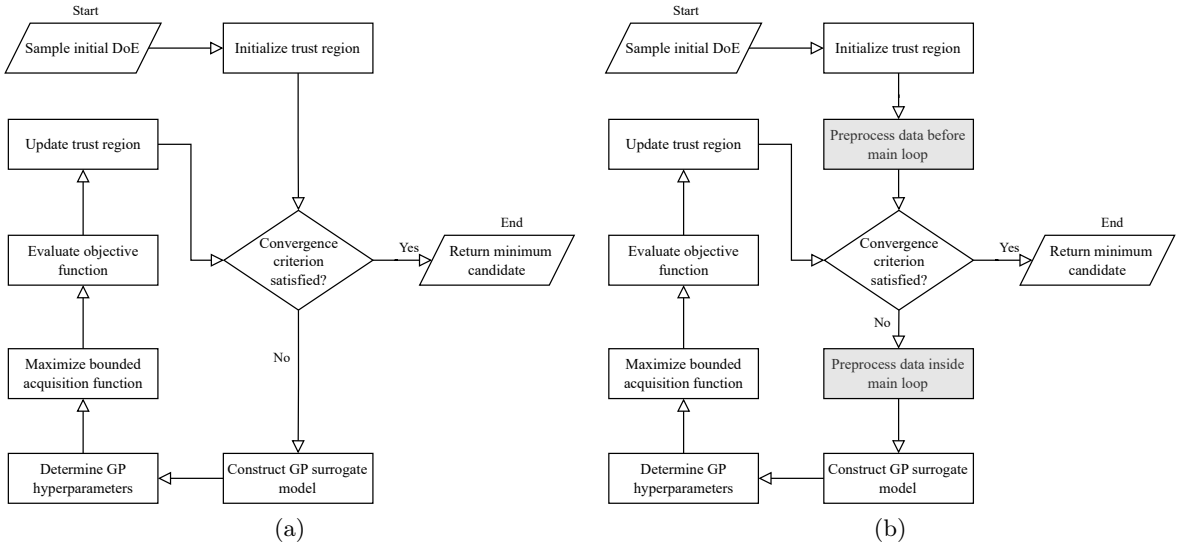



Figure 7.1: A flowchart of (a) a general trust-region-based BO algorithm and (b) one with added data preprocessing steps.

A quintessential example of data preprocessing can also be found in the technique known as min-max normalization [89], which, when applied to the observed output values $Y$, can be given by[1]

$$Y \mapsto \left\{ \frac{y_i - y_{\min}}{y_{\max} - y_{\min}} \,\middle|\, y_i \in Y \right\}. \tag{7.1}$$

This transformation ensures that the current maximum and minimum observed values ($y_{\max}$ and $y_{\min}$) are mapped to 1 and 0, respectively. This operation can be restated as performing a

[1] As in Footnote 3 of Section 5.1, anonymous function notation is used in this section to reduce symbolic overhead by avoiding the need to explicitly name the transformation.

transformation such that some invariant property holds for the transformed data. For min-max normalization, the associated invariant property of Equation 7.1 can be defined as

(i) The current minimum observed output value $y_{\min}$ and the maximum observed value $y_{\max}$ are transformed to be equal to 0 and 1, respectively

$$y_{\min} \mapsto 0 \quad \text{and} \quad y_{\max} \mapsto 1.$$

Using this formulation of preprocessing as a transformation according to an invariant property, another standard preprocessing technique, known as the centring transform, can be restated. This transformation ensures that the current observed minimum input value $\mathbf{x}_{\min}$ (the input associated with the current observed minimum value $y_{\min}$) is at the origin. This transformation can be given by

$$X \mapsto \left\{ \mathbf{x}_i - \mathbf{x}_{\min} \,\middle|\, \mathbf{x}_i \in X \right\}, \tag{7.2}$$

similar to the numerator found in the min-max transform in Equation 7.1, with the associated invariant property

(ii) The minimum observed input value $\mathbf{x}_{\min}$ is transformed to be at the origin

$$\mathbf{x}_{\min} \mapsto \begin{bmatrix} 0 & \dots & 0 \end{bmatrix}^\top.$$

This formulation of transforming the observed data $X$ and $Y$ according to some invariant properties will also be used throughout the rest of this dissertation with 4 invariant properties enumerated as (i)–(iv). It is important to note that these invariant properties are also not invariant in the truest sense, that is to say, that they will always be true necessarily. Rather, they are properties that are *made to be true* or *enforced* through the transformation of the data using the associated transforms.

## 7.2 Rotation Transformation Definition

As stated in the introduction of this chapter, an addition of a rotational component to the trust region used by trust-region-based BO methods, similar to those used in several, classical trust-region-based methods, would be desirable. Unfortunately, the problem of black-box optimization precludes the use of the Jacobian and Hessian of the objective function (as used in classical trust-region-based methods) to determine the directions in the objective function for the rotation of the trust region. Another method to determine the desired arbitrary directions in the objective function is to use the factor analysis distance proposed by Rasmussen and Williams [30, Ch. 5], as discussed in Section 3.2. However, using the full factor analysis distance would be prohibitively expensive due to the excessive number of hyperparameters that need to be optimized at each algorithm iteration.

As shown previously in Section 5.2 and Figure 5.3, the weighted principal components calculated using the observed inputs of the objective function (with each input weighted by the

corresponding observed output value) can describe the underlying structure of the objective function. By rotating the observed data to align these weighted principal components with the coordinate axes, the automatic relevance determination used in the Gaussian process (GP) surrogate of the LABCAT algorithm (which is, crucially, limited to the coordinate axes) can model the objective function more accurately, without the expensive calculation of the full factor analysis distance. As stated in the introduction of this chapter, using a trust region in conjunction with this rotated data can also allow the trust region to effectively expand and contract along local axes of separability, such as diagonal valleys. With the previously established formulation using invariant properties of the previous section, the invariant property that describes this novel rotation can be defined as

(iii) The weighted principal components, described by the columns of the orthogonal rotation matrix $\mathbf{U}$, of the observed input data (centred on the current minimum candidate), with more weight given to input values with a lower corresponding output value, are transformed to be aligned with the coordinate axes (up to reflection)

$$\mathbf{U} \mapsto \operatorname{diag}(\pm 1, \ldots, \pm 1).$$

To enforce this invariant property, the input data is rotated using the transformation

$$\mathbf{X}_{\odot} \mapsto \mathbf{U}^{\top}\mathbf{X}_{\odot}, \tag{7.3}$$

where the centred input data $\mathbf{X}_{\odot}$ is rotated using the transpose of the matrix $\mathbf{U}$ describing the weighted principal components of this data. Note that, for notational convenience, in this dissertation the set of observed inputs $X$ (and transformed representations thereof) can be collected into and decomposed from a matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, where the $i^{\text{th}}$ column of $\mathbf{X}$ corresponds to $\mathbf{x}_i \in X$ and the subscript $\odot$ is appended to input sets or associated matrices to indicate that the input data has been centred using the minimum candidate $\mathbf{x}_{\min}$ from invariant property (ii).

The weighted principal components matrix $\mathbf{U}$ is calculated similarly[2] to the generalized decomposition of Greenacre [82] defined in Equation 5.13 of Section 5.2. Using the SVD, this decomposition is given by

$$\mathbf{X}_{\odot}\mathbf{W} = \mathbf{U}\Sigma\mathbf{V}^{\top}, \tag{7.4}$$

where the general dimensional and sample weight matrices $\mathbf{\Psi}$ and $\mathbf{\Phi}$ are set to the identity matrix and diagonal weight matrix $\mathbf{W}$, respectively. In this decomposition, the columns of the matrix $\mathbf{U}$ correspond to the weighted principal components and, given that $\mathbf{X}_{\odot}$ and $\mathbf{W}$ are matrices with real entries, this rotation matrix is also guaranteed to be orthogonal.

*Proof.* We must prove that, using Equation 7.3, the resulting transformed, centred input data $\mathbf{U}^{\top}\mathbf{X}_{\odot}$ of the transformation from Equation 7.3 satisfies invariant property (iii). That is to say, the weighted principal components of the resulting transformed data are aligned with the coordinate axes. Starting with the observed input data $\mathbf{X}$, the data can be centred

[2]Note the subtle distinction in the calculation of these weighted principal components which uses data centred on the *minimum candidate solution* instead of the *mean of the observed input values* that would often be used in traditional PCA.

using the current minimum candidate $\mathbf{x}_{\min}$ using a matrix-based form of Equation 7.2, or

$$\mathbf{X}_{\odot} = \mathbf{X} - \mathbf{x}_{\min}\mathbf{1}_n^{\top}. \tag{7.5}$$

The weighted principal components of the observed input data $\mathbf{U}$ is obtained using the SVD from Equation 7.4, given by

$$\mathbf{X}_{\odot}\mathbf{W} = \mathbf{U}\Sigma\mathbf{V}^{\top} \tag{7.6}$$

Substituting the transformed value of $\mathbf{X}_{\odot}$ (i.e., $\mathbf{U}^{\top}\mathbf{X}_{\odot}$) into the SVD of $\mathbf{X}_{\odot}\mathbf{W}$, noting that the rotation matrix $\mathbf{U}$ obtained through the SVD is orthogonal ($\mathbf{U}^{-1} = \mathbf{U}^{\top}$) given that $\mathbf{X}_{\odot}$ and $\mathbf{W}$ are real matrices, simplifying yields

$$\begin{aligned} \therefore \mathbf{U}^{\top}\mathbf{X}_{\odot}\mathbf{W} &= \mathbf{U}^{\top}\mathbf{X}_{\odot}\mathbf{W} \\ &= \mathbf{U}^{\top}\mathbf{U}\Sigma\mathbf{V}^{\top} \\ &= \mathbf{I}\Sigma\mathbf{V}^{\top}. \end{aligned} \tag{7.7}$$

This statement is essentially the definition of the SVD for the matrix product $\mathbf{U}^{\top}\mathbf{X}_{\odot}\mathbf{W}$. Since $\mathbf{U}^{\top}\mathbf{X}_{\odot}$ is the transformed, centred input data, this implies that the weighted principal components of the transformed, centred input data are equal to the identity matrix $\mathbf{I}$. This satisfies invariant property (iii), since

$$\mathbf{I} \in \operatorname{diag}(\pm 1, \ldots, \pm 1). \tag{7.8}$$

In other words, the weighted principal components of the transformed, centred input data $\mathbf{U}^{\top}\mathbf{X}_{\odot}$ are aligned with the coordinate axes, concluding the proof. □

While the choice of weights for the sample-wise weight matrix $\mathbf{W}$ is arbitrary, as noted in Section 5.2, sensible values for these weights for the purpose of optimization can be determined using the observed values $Y$. Assuming that the observed output values have been min-max normalized using Equation 7.1, weights that are biased toward lower output values[3] can easily be calculated by subtracting each element of $Y$ from 1 and aggregating into a diagonal matrix

$$\mathbf{W} = \operatorname{diag}(1 - y_1, 1 - y_2, \ldots, 1 - y_n). \tag{7.9}$$

When comparing the weight matrix $\mathbf{W}$ to the general sample-wise matrix $\mathbf{\Phi}^{\frac{1}{2}}$ of Equations 5.13 and 5.15, it is clear that this choice of weights implies that each observation is weighted by the *square* of the diagonal values of $\mathbf{W}$ instead of linearly (as would be the case if the diagonal values of $\mathbf{W}$ are equal to $\sqrt{1 - y_i}$), adding additional bias to smaller output values. Ideally, this additional bias towards observations with lower output values, which would therefore have higher weights and a larger influence on the weighted principal components, assists in uncovering promising search directions in the objective function (in effect, the weighted principal components). This choice of weights is also verified experimentally in the ablation study performed in Chapter 10.

[3]Weights are biased toward lower output values as this dissertation is focused on minimization. In the context of maximization, the normalized values of $Y$ could be used as weights directly.

This novel rotation transformation is also superficially similar to the whitening transform defined in Equation 5.11 (minus the rescaling by the singular values), with the weighted observed inputs now also decorrelated. An example demonstrating this novel rotation using sample data drawn from an arbitrary, ellipsoidal function is given in Figure 7.2.

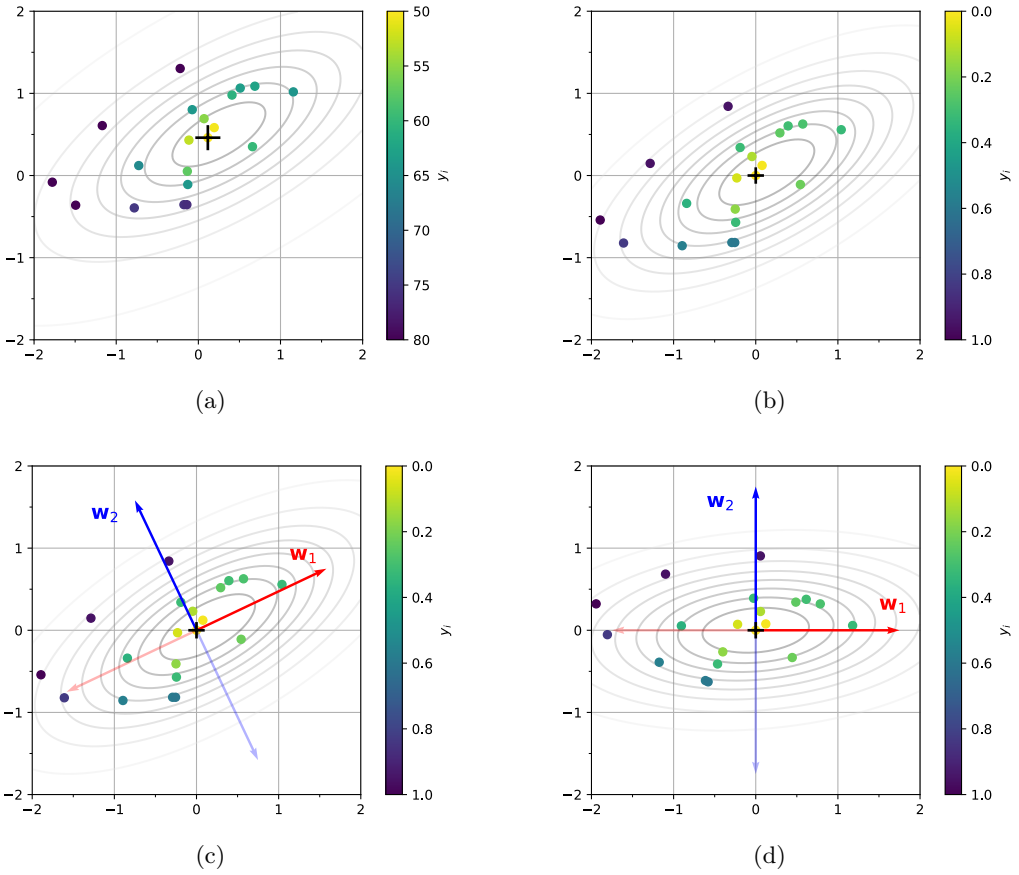
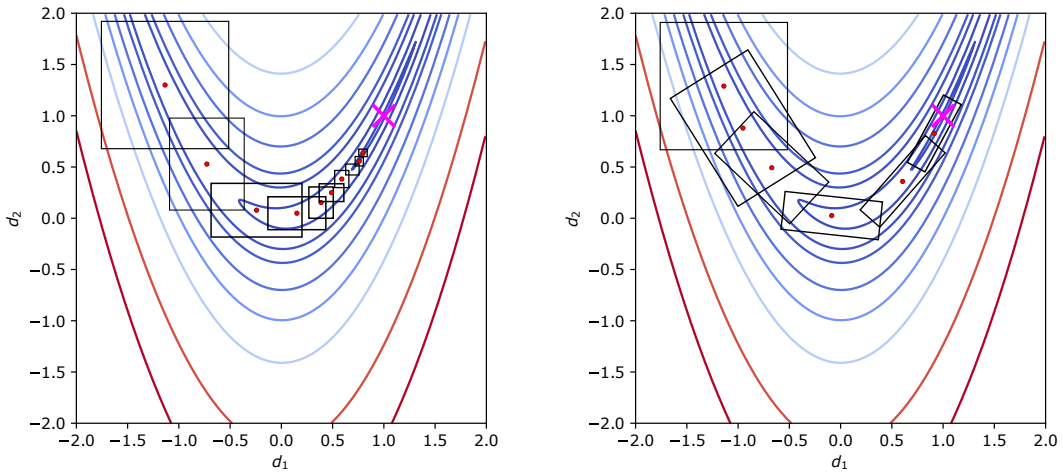



Figure 7.2: A visualization demonstrating an example of enforcing the invariant properties (i), (ii) and (iii) on (a) a number of observations from an arbitrary function, where the observed output values are represented using a colour map. The data is (b) centred on the minimum candidate (marked with a +), the output values are min-max normalized and (c) the weighted principal components $\mathbf{w}_1$ and $\mathbf{w}_2$ are shown. Using these principal components, (d) the input data is rotated such that these principal components are aligned with the coordinate axes, with all of the invariant properties now enforced.

It is also important to note that this transform, as well as the min-max and centring transforms, is invertible, meaning that the original data can be recovered from the transformed data (such as for the centred data $\mathbf{X}_{\odot} \leftrightarrow \mathbf{X}$) or that conversions can be performed between the original and transformed space for new points.

## 7.3 Illustrative Example

To illustrate the advantage of this rotation strategy compared to other trust-region-based BO methods, consider the optimization of the Rosenbrock function [58], a well-known test function with a narrow, banana-shaped valley leading towards the global optimum. The starting point for the optimization run is chosen at the end of this valley, typically a very challenging starting point for most optimization algorithms. To visualize the intuition underpinning the trust region rotation and provide a conceptual motivation for this strategy, this optimization scenario is presented in Figure 7.3. Note that this is not meant to be understood as a quantitative or qualitative result, but rather as an illustration of expected and desired behaviour.




(a) No rotation

(b) Rotation included

Figure 7.3: Illustrative example showing a typical run of a hypothetical trust-region-based BO algorithm (with independent side lengths) applied to the 2-D Rosenbrock function (a) without and (b) with weighted-principal-component-based trust region rotation. A subset of trust regions (indicated in black) centred on the respective minimum candidate solutions (indicated in red) are given, the global optimum is indicated by the magenta cross at $(1.0, 1.0)$ and observations other than the minimum candidate are not indicated to maintain visual clarity.

Firstly, consider a hypothetical trust-region-based BO algorithm, as described in Algorithm 3 and the flowchart of Figure 7.1 (a), with some trust region update strategy that can independently update the trust region side lengths and without the weighted principal component rotation, given in Figure 7.3 (a). This algorithm is expected to behave similarly to other trust-region-based algorithms, with the trust region recentred on new minimum candidates and moving along the valley. In the areas of the valley aligned with the coordinate axis, such as near the origin, the trust region is allowed to expand along the valley. However, the narrowness of the valley near the minimum that is not aligned with the coordinate axes forces the trust region to shrink rapidly. Due to the constraint placed on the acquisition function that the objective function may only be evaluated at subsequent points *inside* this trust region, this premature contraction slows progress towards the optimum considerably.

Next, consider the same hypothetical trust-region-based BO algorithm with the addition of the weighted-principal-component-based rotation at the beginning of each algorithm (in effect, performing the first algorithm in the sequentially updated space of the transformed data), similar to Figure 7.1 (b). By inspecting Figure 7.3 (b), it is clear that this rotation yields a clear improvement. The rotation aligns the major axis of the rectangular trust region along the valley of the Rosenbrock function as the trust region moves through the valley. The trust region can now exploit the local separability of the valley by expanding and contracting along the major and minor axes of the trust region. This version of the algorithm finds the optimum more efficiently than in Figure 7.3 (a), demonstrating the potential value of the principal component rotation.

In summary, this chapter has presented the novel preprocessing step of a weighted-principal-component-based rotation in a trust-region-based BO framework to allow the trust region to adapt to local separability in the objective function, such as curved or diagonal valleys. The description of this data preprocessing step as a transformation of the observed data with an associated invariant property will also be used in the following chapter describing a length-scale-based rescaling of the observed data.

# Chapter 8

# Length-scale-based Rescaling

During the course of any trust-region-based Bayesian optimization (BO) algorithm (as discussed in Section 4.2), the size of the trust region necessarily shrinks as the algorithm converges to a solution. This shrinking trust region may lead to observations being clustered together or ill-conditioned, causing the spatial covariance matrix $\mathbf{K}$ to become near singular, as noted in Section 1.1. To prevent this, a transformation is proposed that transforms the observations using a novel, local-length-scale-based rescaling. Using the transformed representation of the observed input data, subsequent actions (such as determining the next point to sample) are performed that work well for a much smaller range of objective function values compared to the original space. In this transformed space, the observations ideally remain well-conditioned and -distributed, even if the corresponding points in the objective function space are not. This transformation, in turn, also induces a trust region update strategy based on the local length-scales, allowing superior performance for separable and ill-conditioned objective functions.

Using the form established in the previous chapter of a transformation applied to the observed data with an associated invariant property enforced on the transformed data, this chapter defines the novel rescaling of the observed input values using the most likely length-scales, specifically for the case of a Gaussian process (GP) fitted to the observed data using a squared exponential kernel with automatic relevance determination.

## 8.1 Rescaling Transformation Definition

As described in Algorithm 3 and illustrated in Figure 7.1, the main loop of trust-region-based BO consists of constructing a GP surrogate to approximate the objective function using previously observed values and selecting the next point to observe by using the GP to maximize an acquisition function bounded by a trust region. In the previous chapter, it was also shown that the observed data can be preprocessed at the start of the loop before the GP is constructed in order to improve the accuracy of the resulting surrogate model. However, even though this preprocessing may allow the GP to better *model* the objective function (for example, by using the weighted principal component rotation of the previous chapter), problems may still arise when trying to maximize an acquisition function with this surrogate model. This is especially the case for ill-conditioned objective functions, where the acquisition function can exhibit high sensitivity in specific directions. Additionally, as the trust region shrinks or observations become clustered, numerical instability can arise in the GP due to a nearly singular spatial

covariance matrix $\mathbf{K}$, as noted in Section 1.1.

To address these challenges, the observed inputs are rescaled according to the local smoothness (in effect, the sensitivity) of the GP surrogate for each dimension in order to achieve a more uniform smoothness (sensitivity) in each dimension. As described in Section 3.2, an elegant measure of the local smoothness of the GP can be found in the length-scale parameter $\ell$ of the squared exponential kernel function, with automatic relevance determination extending this parameter to individual, independent values for each dimension $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_d)$. This length-scale parameter can be estimated from the observed data (i.e., $\hat{\boldsymbol{\ell}}$) by maximizing the log marginal likelihood (Equation 3.13) and can therefore be considered as the *inferred local smoothness* of the objective function and be used in the rescaling process. In essence, this rescaling reduces the risk of the acquisition function becoming overly sensitive to local fluctuations or ill-conditioning, thereby enhancing the stability and reliability of the optimization process.

While this rescaling cannot be considered as a preprocessing step for the GP (as it is performed after the GP is constructed and the length-scales are inferred), it can be considered as an additional data preprocessing step for the maximization of the acquisition function. As such, the description of data preprocessing from Section 7.1 as a transformation applied to the observed data with an associated invariant property can be used. For the desired length-scale-based rescaling, the invariant property can be given by

(iv) The most likely length-scales $\hat{\boldsymbol{\ell}}$ for a GP that has been constructed with the observed inputs, using a squared exponential kernel with automatic relevance determination, are rescaled to unity, or

$$\hat{\boldsymbol{\ell}} \mapsto (1, 1, \ldots, 1).$$

Using this invariant property, the transformation applied to the observed inputs $X$, using the most likely length-scales of a GP using a squared exponential kernel with automatic relevance determination $\hat{\boldsymbol{\ell}}$ calculated using Equation 3.16, is given by

$$\mathbf{X} \mapsto \hat{\mathbf{L}}^{-1}\mathbf{X}, \tag{8.1}$$

where the most likely length-scales are collected into a diagonal scaling matrix

$$\hat{\mathbf{L}}^{-1} = \operatorname{diag}(\hat{\ell}_1, \hat{\ell}_2, \ldots, \hat{\ell}_n)^{-1}. \tag{8.2}$$

An example of this length-scale-based rescaling can be seen in Figure 8.1.

Note that, for the new GP surrogate (if using the squared exponential kernel) with the transformed data and unit length-scales, there is no need to recalculate the kernel matrix $\mathbf{K}$ of the GP constructed using the original input data, as both the observed inputs and length-scales have been scaled by the same factor.

*Proof.* Given a GP constructed using a set of observed inputs $X$ and the most likely length-scales $\hat{\boldsymbol{\ell}}$, we must prove that this is equivalent to another GP constructed with the rescaled input values $\hat{\mathbf{L}}^{-1}\mathbf{X}$ and unit length-scales. Given that the set of observed outputs remain unchanged, the GPs can be parameterized by their respective kernel matrices $\mathbf{K}$, the entries

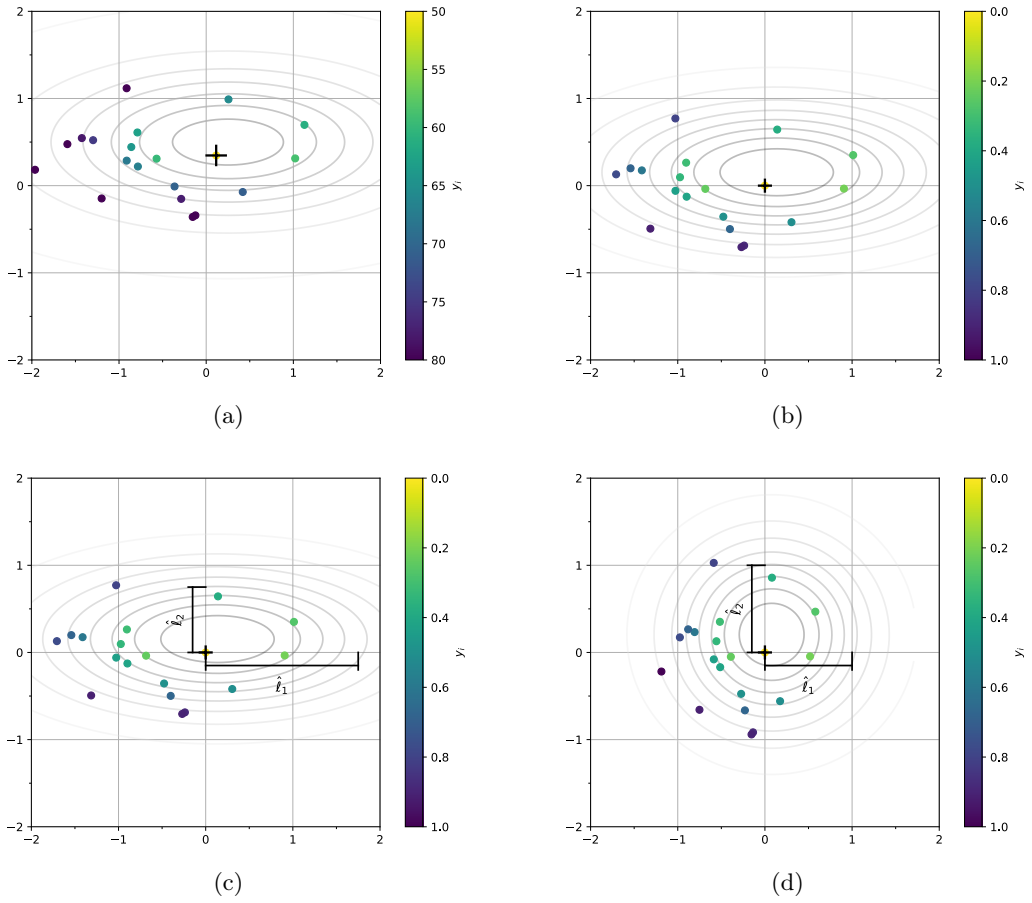



Figure 8.1: A visualization demonstrating an example of enforcing the invariant properties described by (i), (ii), and (iv) on (a) a number of observations from an arbitrary, mildly ill-conditioned and separable function, where the observed output values are represented using a colour map. The data is first (b) centred on the minimum candidate (marked with a +), the output values are normalized and (c) the most likely length-scales ($\hat{\ell}_1$, $\hat{\ell}_2$) for a GP fitted to the data are shown. Using these length-scales, the observed input data is (d) rescaled such that these length-scales equal unity, with all invariant properties now preserved.

of which are, in turn, defined by the respective kernel functions. For the given GP using $X$ and $\hat{\boldsymbol{\ell}}$, the kernel function (chosen as the squared exponential function with automatic relevance determination from Equation 3.9 in this dissertation) is given by

$$k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j \,; \hat{\boldsymbol{\ell}}) = \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \hat{\boldsymbol{\Lambda}}^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right) + \sigma_n^2 \delta_{ij} \tag{8.3}$$
$$\text{where} \quad \hat{\boldsymbol{\Lambda}} = \text{diag}(\hat{\ell}_1^2, \hat{\ell}_2^2, \ldots, \hat{\ell}_d^2).$$

The diagonal factor $\hat{\boldsymbol{\Lambda}}$ in this kernel (i.e., the axis-aligned length-scales of the automatic

relevance determination) can be factorized as

$$\begin{aligned} \hat{\mathbf{\Lambda}}^{-1} &= (\hat{\mathbf{L}}\,\hat{\mathbf{L}})^{-1} \\ &= \hat{\mathbf{L}}^{-1}\hat{\mathbf{L}}^{-1} \\ \text{where} \quad \hat{\mathbf{L}} &= \operatorname{diag}(\hat{\ell}_1, \hat{\ell}_2, \ldots, \hat{\ell}_d) \end{aligned} \tag{8.4}$$

and simplifying the kernel with this factorization yields

$$\begin{aligned} \therefore\therefore\therefore k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j\,;\hat{\boldsymbol{\ell}}) &= \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \hat{\mathbf{L}}^{-1}\hat{\mathbf{L}}^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right) + \sigma_n^2\delta_{ij} \\ &= \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\hat{\mathbf{L}}^{-1}\mathbf{x}_i - \hat{\mathbf{L}}^{-1}\mathbf{x}_j)^\top \mathbf{I}^{-1}(\hat{\mathbf{L}}^{-1}\mathbf{x}_i - \hat{\mathbf{L}}^{-1}\mathbf{x}_j)\right) + \sigma_n^2\delta_{ij}. \end{aligned} \tag{8.5}$$

It is clear that this rescaled kernel is now equivalent to a squared exponential kernel using the rescaled input data $\hat{\mathbf{L}}^{-1}\mathbf{X}$ with unit length-scales

$$\therefore k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j\,;\hat{\boldsymbol{\ell}}) \iff k_{\text{SE}}(\hat{\mathbf{L}}^{-1}\mathbf{x}_i, \hat{\mathbf{L}}^{-1}\mathbf{x}_j\,;\mathbf{1}_d). \tag{8.6}$$

Therefore, GPs constructed using these kernel functions would be equivalent, since entries defining the kernel matrix $\mathbf{K}$ would be equal, or

$$\mathbf{K} = [k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j\,;\hat{\boldsymbol{\ell}})]_{1\leq i,j\leq n} = [k_{\text{SE}}(\hat{\mathbf{L}}^{-1}\mathbf{x}_i, \hat{\mathbf{L}}^{-1}\mathbf{x}_j\,;\mathbf{1}_d)]_{1\leq i,j\leq n}, \tag{8.7}$$

concluding the proof. □

Next, to see the influence of the length-scale-based rescaling on the use of a trust region in a trust-region-based BO framework, it is important to restate the observations that (a) the inferred length-scale parameter $\hat{\boldsymbol{\ell}}$ of the squared exponential kernel function in a Gaussian process (GP) with automatic relevance determination is roughly proportional to the smoothness of the objective function in each dimension, and that (b) the inferred length-scales for a cluster of observations tend to be smaller as a cluster of observations becomes smaller. By rescaling the observed inputs using these inferred local length-scales while keeping a trust region with a fixed size, the size of an *inferred* trust region in the original objective function space can be manipulated to account for the local smoothness of the objective function. This rescaling can, therefore, form the basis of a trust region update strategy by expanding the space of observations in regions of the objective function with longer length-scales (in other words, smoother regions of the objective function) and vice versa.

It is important to note that using this length-scale-based rescaling in isolation, without discarding observations or using a subset of observations to keep the cluster small, would simply lead to rescaling by the global length-scales of the objective function (in effect, the underlying global smoothness). While this would essentially be a form of dynamic preconditioning without prior knowledge of the input sensitivities and may be advantageous for the maximization of the acquisition function for separable, ill-conditioned objective functions, eventually the same numerical issues as standard BO will be encountered that would inhibit convergence. Keeping the active cluster of observations small would encourage the length-scales to be smaller,

allowing convergence to a solution through the repeated rescaling using the length-scales of the shrinking clusters as the algorithm progresses.

In summary, by performing the length-scale-based rescaling at each algorithm iteration in a traditional trust-region-based BO framework, this transformation can allow for improved convergence characteristics and better performance on select separable or ill-conditioned functions, as seen in the ablation study of Chapter 10. However, the full potential of the rescaling is harnessed when combined with the trust-region rotation of the previous chapter, extending the improved performance to more general objective functions. This combination is used by the LABCAT algorithm, detailed in the next chapter.

# Chapter 9

# Detailed Description of the LABCAT Algorithm

**ἄλγος** \ál.gos\: **I.** *pain* of body, **Il.**, **Soph.**
**2.** *pain* of mind, *grief*, *distress*, **Hom.**
**II.** *anything that causes pain*, **Bion.**, **Anth.**

— H. G. Lidell & R. Scott, *A Greek-English Lexicon*

This chapter presents a detailed, mathematical description of the components of the proposed LABCAT algorithm, as outlined in Chapter 6. This chapter begins by presenting a combined observation transformation using the transforms defined in Chapter 7 and 8. Furthermore, the iterative calculation of the parameters of this combined transformation and the approximative estimation of the most-likely Gaussian process (GP) hyperparameters for the transformed data are also given. Additionally, this chapter describes the specific implementation of the trust region, observation discarding strategy, algorithm initialization and termination used in the LABCAT algorithm. The chapter concludes with a detailed mathematical description and computational complexity analysis of the algorithm, synthesizing the work previously discussed.

## 9.1 Combined Observation Transformation

In the flowchart of Figure 6.1 describing the different steps of the LABCAT algorithm, it can be seen that the main loop of the algorithm contains several steps to transform the observed data from the objective function. Specifically, the observed outputs are min-max normalized and the observed inputs are then recentred, rotated about the current minimum using the weighted principal components and finally rescaled using the most likely length-scales. As stated in Chapters 7 and 8, these preprocessing steps can each be described by some transformation applied to the observed data with an associated invariant property. In order to combine the effects of each of these transformations, the LABCAT algorithm maintains both the original observations and a transformed representation thereof which ensures that all of the invariant properties (i)–(iv) hold, leveraging the combined advantages of the transformations as previously described in the respective sections.

In order to facilitate the transformation of new points between the original space of the objective function and the space of the transformed observations, an explicit mapping is defined between the original and transformed observations using an affine transformation. The parameters of this transformation encode the relationship between the two sets of observations and allow representations of a new point in one to be calculated for the other. The remainder of this section firstly defines the combined transform and the associated transformation parameters before defining functions to iteratively update the transformation without a full recalculation of the transformation parameters.

Note that, in a slight abuse of notation to ensure readability, for the rest of this chapter a tuple of an observed input set $X$ and an output set $Y$ (or transformations thereof) is used to indicate a dataset $(X, Y) = \mathcal{D} = \{(\mathbf{x}_i, y_i) \,|\, i \in \{1, \ldots, n\}\}$ in order to simplify the construction of a GP and associated acquisition function, in addition to the continued use of $\mathbf{X} \in \mathbb{R}^{d \times n}$ and $\mathbf{y} \in \mathbb{R}^d$ (and transformed representations thereof) to indicate the matrix composed of the set of observed inputs $X$ and vector composed of the set of observed outputs $Y$.

### 9.1.1 Transformation Definition

The combined, full-rank affine transformation is constructed from the translational, rotational and scaling components of the centering, weighted-principal-component rotation and length-scale-based rescaling transforms, respectively, defined in Equations 7.2, 7.3 and 8.1. This combined transformation defined in terms of the transformed representations of the observed inputs, indicated using the primed counterpart of a variable, is given by

$$\mathbf{x}_i = \mathbf{R}\mathbf{S}\mathbf{x}'_i + \mathbf{c} \;\; \forall i \in \{1, \ldots, n\}, \tag{9.1}$$

which can also be expressed using the matrices

$$\mathbf{X} = \mathbf{R}\mathbf{S}\mathbf{X}' + \mathbf{c}\mathbf{1}_n^\top, \tag{9.2}$$

with the offset vector $\mathbf{c}$, rotation matrix $\mathbf{R}$ and scaling matrix $\mathbf{S}$ calculated to ensure that the invariant properties (ii), (iii) and (iv), respectively, hold for the transformed observed inputs $X'$.[1] Similarly, a second mapping is defined between the original and transformed observed output values, given by

$$y_i = a y'_i + b \;\; \forall i \in \{1, \ldots, n\}, \tag{9.3}$$

which can also be expressed using the vectors

$$\mathbf{y} = a\mathbf{y}' + b\mathbf{1}_n^\top, \tag{9.4}$$

where the two scalar transformation parameters $a$ and $b$ are calculated such that $Y'$ is min-max normalized (invariant property (i)). This transformed representation can be visualized by following the operations performed on a set of data from an arbitrary, ellipsoidal function in Figure 7.2 with the rescaling performed in Figure 8.1 (c) and (d).

[1] For mathematical convenience, the relationship between $X$ and $X'$ is defined in terms of a transformation applied to the transformed points. Since the transformation is invertible, the relationship can equivalently be stated as the transformation applied to the original points $\mathbf{X}' = \mathbf{S}^{-1}\mathbf{R}^\top(\mathbf{X} - \mathbf{c}\mathbf{1}_n^\top)$.

In general, the transformation parameters of these two transformations may not be unique (such as the rotation of invariant property (iii) defined up to reflection). By inspecting the individual transforms applied to the observed input data associated with invariant properties (ii)–(iv) from Equations 7.2, 7.3 and 8.1, it is clear that these equations are purely translational, rotational, and scaling operations, respectively. As such, a naïve choice for the rotational component $\mathbf{R}$, scaling component $\mathbf{S}$ and translational component $\mathbf{c}$, based on these equations and the observed values $X$, can be given by

$$\mathbf{R} = \mathbf{U}, \quad \mathbf{S} = \hat{\mathbf{L}}, \quad \text{and} \quad \mathbf{c} = \mathbf{x}_{\min}, \tag{9.5}$$

Similarly, choices for the scaling and offset components of the transformation applied to the observed output data $a$ and $b$, based on the min-max normalization of Equation 7.1, can be given by

$$a = y_{\max} - y_{\min} \quad \text{and} \quad b = y_{\min}. \tag{9.6}$$

While these choices are sufficient[2] for the ideal, mathematical case with infinite precision, the implementation may be precluded by the finite-precision of real-world computing machines. Specifically, calculating these values in practice requires operations involving the original input points $\mathbf{X}$, which, as established in the previous chapters, may become clustered during the course of a BO algorithm. This may lead to numerical issues due to the ill-conditioning of this matrix and cause values such as the weighted principal components $\mathbf{U}$ to be impossible to calculate directly. As such, the next section defines the transformation parameters in terms of recurrence relationships using the transformed representations of the observed data and the values of the transformation parameters from the previous algorithm iteration, avoiding the use of the original data directly.

### 9.1.2 Iterative Transformation Parameter Calculation

During the execution of the LABCAT algorithm, it is not necessary to perform a full recalculation of $X'$ and $Y'$ from the current values of $X$ and $Y$ (with the respective transformation parameters defined in the input and output transformation definitions from Equations 9.2 and 9.3) at each algorithm iteration. Instead, to calculate the values for the current iteration, indicated by the subscript $t$, the transformed representations obtained from the preceding algorithm iteration, denoted as $X'_{t-1}$ and $Y'_{t-1}$, are leveraged for this calculation. Moreover, the associated transformation parameters of the previous algorithm iteration are also retained, also indicated by the subscript $t-1$. To provide further context for this process in the LABCAT algorithm and introduce the necessary notation, a reduced form of the flowchart describing the overall LABCAT algorithm is given in Figure 9.1.

As seen in Figures 6.1 and 9.1, the sets of observed data at the end of an algorithm iteration $\tilde{X}'_{t-1}$ and $\tilde{Y}'_{t-1}$ may incorporate or remove observations such that the invariant properties (i)–(iv) no longer hold for the previously calculated transformed representations $X'_{t-1}$ and $Y'_{t-1}$ at the start of the algorithm iteration. To efficiently restore the invariant properties for $X'_t$ and $Y'_t$, these values, with the associated transformation parameters, are calculated in terms of the

[2] Proofs showing that these choices of transformations parameters ensure that the invariant properties (i)–(iv) hold for $X'$ and $Y'$ can optionally be found in Appendix A, as these choices do not form part of the iterative scheme of the next section.

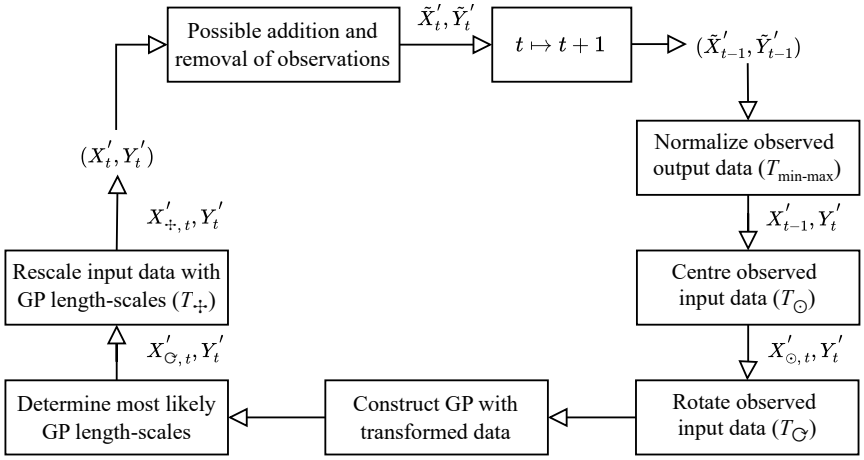



Figure 9.1: Reduced flowchart based on the main loop of Figure 6.1 with specific focus on the iterative transformation parameter calculation process.

transformation defined in the previous algorithm iteration and the sets of observed values at the *end* of the previous algorithm iteration. Note that, to define a proper recurrence relation, the initial values $\tilde{X}'_0$ and $\tilde{Y}'_0$ used by the LABCAT algorithm, with associated transformation parameters, are calculated using a modified version of the transformations from Equations 9.1 and 9.3 based on the bounds of the objective function $\Omega$ and is given in Section 9.5.

As shown in the flowchart of Figure 9.1, the transformed outputs of the end of the previous iteration $\tilde{Y}'_{t-1}$ are firstly renormalized using the minimum and maximum values of this set, $\tilde{y}'_{\min,\,t-1}$ and $\tilde{y}'_{\max,\,t-1}$, to ensure that invariant property (i) holds, given by

$$Y'_t = T_{\updownarrow,\,t}(\tilde{Y}'_{t-1}) := \left\{ \frac{\tilde{y}'_{i,\,t-1} - \tilde{y}'_{\min,\,t-1}}{\tilde{y}'_{\max,\,t-1} - \tilde{y}'_{\min,\,t-1}} \;\middle|\; \tilde{y}'_{i,\,t-1} \in \tilde{Y}'_{t-1} \right\} \tag{9.7}$$

and the transformation parameters in the output transformation from Equation 9.3 are updated by

$$a_t = a_{t-1} \cdot (\tilde{y}'_{\max,\,t-1} - y'_{\min,\,t-1}), \tag{9.8}$$
$$b_t = b_{t-1} + a_{t-1} \cdot \tilde{y}'_{\min,\,t-1}. \tag{9.9}$$

*Proof.* At the start of some algorithm iteration $t-1 > 0$, the transformation from the observed outputs $Y_{t-1}$ to the transformed outputs $Y'_{t-1}$ is given by

$$y_{i,\,t-1} = a_{t-1} \cdot y'_{i,\,t-1} + b_{t-1}, \tag{9.10}$$

with $a_{t-1}$ and $b_{t-1}$ calculated such that invariant property (i) holds. Between the calculation of these values and the end of the algorithm iteration, observations may be added to or removed from $Y'_{t-1}$ such that the invariant property may no longer hold for the transform

$$\tilde{y}_{i,\,t-1} = a_{t-1} \cdot \tilde{y}'_{i,\,t-1} + b_{t-1}, \tag{9.11}$$

using the observed values at the end of the algorithm iteration $\tilde{Y}_{t-1}$.

For the next algorithm iteration $t$, we must prove that

$$y_{i,\,t} = a_t \cdot y'_{i,\,t} + b_t \tag{9.12}$$

expresses a new transformation, calculated using $\tilde{Y}'_{t-1}$, $a_{t-1}$ and $b_{t-1}$, such that invariant property (i) holds.

Substituting the updated values of $Y_t$, $a_t$ and $b_t$ from Equations 9.7, 9.8, and 9.9, respectively, into the definition of the transformation at algorithm iteration $t$ from Equation 9.12, simplifying yields

$$\begin{aligned}
y_{i,\,t} &= a_{t-1} \cdot (\tilde{y}'_{\max,\,t-1} - \tilde{y}'_{\min,\,t-1}) \cdot \left( \frac{\tilde{y}'_{i,\,t-1} - \tilde{y}'_{\min,\,t-1}}{\tilde{y}'_{\max,\,t-1} - \tilde{y}'_{\min,\,t-1}} \right) + b_{t-1} + a_{t-1} \cdot \tilde{y}'_{\min,\,t-1} \\
&= a_{t-1} \cdot (\tilde{y}'_{i,\,t-1} - \tilde{y}'_{\min,\,t-1}) + b_{t-1} + a_{t-1} \cdot \tilde{y}'_{\min,\,t-1} \\
&= a_{t-1} \cdot \tilde{y}'_{i,\,t-1} + b_{t-1} - a_{t-1} \cdot \tilde{y}'_{\min,\,t-1} + a_{t-1} \cdot \tilde{y}'_{\min,\,t-1} \\
&= a_{t-1} \cdot \tilde{y}'_{i,\,t-1} + b_{t-1}.
\end{aligned}$$

This result implies that

$$\therefore Y_t = \tilde{Y}_{t-1}, \tag{9.13}$$

which must be true, given that the set of untransformed output observations does not change between the end of the previous algorithm iteration and the calculation of the new transformation parameters at the start of the current algorithm iteration.

Therefore, Equation 9.12 implies a mapping using the same original output points from Equation 9.11 expressed using the new, transformed output values $Y'_t$ and transformation parameters $a_t$ and $b_t$ such that invariant property (i) holds, concluding the proof. □

Having reestablished invariant property (i) for the observed output data, we now turn to the task of reestablishing invariant properties (ii)–(iv) for the observed input data in order. Firstly, invariant property (ii) is preserved by recentring $\tilde{X}'_{t-1}$ on the (possibly new) minimum candidate $\tilde{\mathbf{x}}'_{\min,\,t-1}$ (the minimum candidate solution of $(\tilde{X}'_{t-1}, \tilde{Y}'_{t-1})$) with

$$\mathbf{X}'_{\odot,\,t} = T_{\odot,\,t}(\tilde{X}'_{t-1}) \coloneqq \tilde{\mathbf{X}}'_{t-1} - \tilde{\mathbf{x}}'_{\min,\,t-1}\mathbf{1}_n^\top, \tag{9.14}$$

with the offset vector $\mathbf{c}_t$ updated using the value of the previous iteration and the (possibly new and non-zero) minimum candidate

$$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{R}_{t-1}\mathbf{S}_{t-1}\tilde{\mathbf{x}}'_{\min,\,t-1}. \tag{9.15}$$

Next, to restore invariant property (iii), it can be seen by inspecting the transform defined in Equation 9.2 that multiplying the transformed input data $\mathbf{X}'$ by the scaling matrix $\mathbf{S}$ yields a representation of the data in an intermediate (rotated and translated), affine space of the original input space of $\mathbf{X}$. The weighted principal components $\mathbf{U}_{\text{affine},\,t}$, with the weights $\mathbf{W}_t$ calculated using $Y'_t$ according to Equation 7.9, can be calculated in this intermediate space as

$$\mathbf{S}_{t-1}\mathbf{X}'_{\odot,\,t}\mathbf{W}_t = \mathbf{U}_{\text{affine},\,t}\mathbf{\Sigma}_{\text{affine},\,t}\mathbf{V}^\top_{\text{affine},\,t}. \tag{9.16}$$

This rotation matrix $\mathbf{U}_{\text{affine}}$, after a change of basis using $\mathbf{S}_{t-1}$ from the intermediate space, can be used to rotate $\mathbf{X}'_{\odot,\,t}$ using the transform

$$\mathbf{X}'_{\complement,\,t} = T_{\complement,\,t}(\mathbf{X}'_{\odot,\,t}) \coloneqq \mathbf{S}_{t-1}^{-1}\mathbf{U}_{\text{affine},\,t}^{\top}\mathbf{S}_{t-1}\mathbf{X}'_{\odot,\,t} \tag{9.17}$$

and calculate the transformation parameter $\mathbf{R}_t$ in terms of the value from the previous iteration

$$\mathbf{R}_t = \mathbf{R}_{t-1}\mathbf{U}_{\text{affine},\,t}, \tag{9.18}$$

essentially updating the rotational component of the transformation defined in Equation 9.1 without requiring the full transformation of $X'$ back to $X$, which would require additional matrix multiplications.

Finally, similarly to the rescaling performed in Equation 8.1, the most likely length-scales $\hat{\boldsymbol{\ell}}$ for a GP fitted to $\mathbf{X}'_{\complement,\,t}$ and $Y'_t$ with an automatic relevance determination kernel (collected into diagonal matrix $\hat{\mathbf{L}}_t$) are used to rescale $\mathbf{X}'_{\complement,\,t}$ according to

$$\mathbf{X}'_{\updownarrow,\,t} = T_{\updownarrow,\,t}(\mathbf{X}'_{\complement,\,t}) \coloneqq \hat{\mathbf{L}}_t^{-1}\mathbf{X}'_{\complement,\,t} \tag{9.19}$$

and recalculate the parameter $\mathbf{S}_t$ in terms of the value from the previous iteration

$$\mathbf{S}_t = \hat{\mathbf{L}}_t\mathbf{S}_{t-1}, \tag{9.20}$$

also essentially updating the scaling component of the transformation defined in Equation 9.1, enforcing invariant property (iv) without requiring the full transformation of $X'$ back to $X$.

By composing the update formulae defined in Equations 9.14, 9.17 and 9.19, a new transformation is implied with an updated representation of the transformed input data and transformation parameters that ensure that invariant properties (ii)–(iv) hold, or

$$\begin{aligned} &\therefore X'_t = T_{\updownarrow,\,t}(\,T_{\complement,\,t}(\,T_{\odot,\,t}(\tilde{X}'_{t-1}))) \\ &\implies \mathbf{X}_t = \mathbf{R}_t\mathbf{S}_t\mathbf{X}'_t + \mathbf{c}_t\mathbf{1}_n^{\top}. \end{aligned} \tag{9.21}$$

*Proof.* At the start of some algorithm iteration $t-1 > 0$, the transformation from the observed inputs $\mathbf{X}_{t-1}$ to the transformed inputs $\mathbf{X}'_{t-1}$ is given by

$$\mathbf{X}_{t-1} = \mathbf{R}_{t-1}\mathbf{S}_{t-1}\mathbf{X}'_{t-1} + \mathbf{c}_{t-1}\mathbf{1}_n^{\top}, \tag{9.22}$$

with $\mathbf{R}_{t-1}$, $\mathbf{S}_{t-1}$ and $\mathbf{c}_{t-1}$ calculated such that the invariant properties (ii)–(iv) hold. Between the calculation of these values and the end of the algorithm iteration, observations may be added to or removed from $X'_{t-1}$ such that the invariant properties may no longer hold for the transform

$$\tilde{\mathbf{X}}_{t-1} = \mathbf{R}_{t-1}\mathbf{S}_{t-1}\tilde{\mathbf{X}}'_{t-1} + \mathbf{c}_{t-1}\mathbf{1}_n^{\top}, \tag{9.23}$$

using the observed values at the end of the algorithm iteration $\tilde{X}_{t-1}$.

For the next algorithm iteration $t$, we must prove that

$$\mathbf{X}_t = \mathbf{R}_t\mathbf{S}_t\mathbf{X}'_t + \mathbf{c}_t\mathbf{1}_n^{\top}, \tag{9.24}$$

expresses a new transformation using new transformation parameters, calculated using Equation 9.21 such that the invariant properties (ii)–(iv) hold for the new set of transformed inputs $\mathbf{X}'_t$.

Substituting the updated values of $\mathbf{X}'_{\updownarrow,t}$ and $\mathbf{S}_t$ from Equations 9.19 and 9.20 into the definition of the transformation at algorithm iteration $t$ from Equation 9.24, given that both $\mathbf{S}$ and $\hat{\mathbf{L}}^{-1}$ are diagonal scaling matrices and that multiplication of diagonal matrices are commutative, simplifying yields

$$\begin{aligned}
\therefore \mathbf{X}_t &= \mathbf{R}_t\mathbf{S}_t\mathbf{X}'_{\updownarrow,t} + \mathbf{c}_t\mathbf{1}_n^\top \\
&= \mathbf{R}_t(\hat{\mathbf{L}}_t\mathbf{S}_{t-1})(\hat{\mathbf{L}}_t^{-1}\mathbf{X}'_{\complement,t}) + \mathbf{c}_t\mathbf{1}_n^\top \\
&= \mathbf{R}_t\mathbf{S}_{t-1}\hat{\mathbf{L}}_t\hat{\mathbf{L}}_t^{-1}\mathbf{X}'_{\complement,t} + \mathbf{c}_t\mathbf{1}_n^\top \\
&= \mathbf{R}_t\mathbf{S}_{t-1}\mathbf{X}'_{\complement,t} + \mathbf{c}_t\mathbf{1}_n^\top.
\end{aligned}$$

Substituting the updated values of $\mathbf{X}'_{\complement,t}$ and $\mathbf{R}_t$ from Equations 9.17 and 9.18 into this transform, recalling that the matrix $\mathbf{U}$ obtained from the SVD is orthogonal ($\mathbf{U}^{-1} = \mathbf{U}^\top$), and simplifying yields

$$\begin{aligned}
\therefore \mathbf{X}_t &= \mathbf{R}_t\mathbf{S}_{t-1}\mathbf{X}'_{\complement,t} + \mathbf{c}_t\mathbf{1}_n^\top \\
&= (\mathbf{R}_{t-1}\mathbf{U}_{\text{affine},t})\mathbf{S}_{t-1}(\mathbf{S}_{t-1}^{-1}\mathbf{U}_{\text{affine},t}^\top\mathbf{S}_{t-1}\mathbf{X}'_{\odot,t}) + \mathbf{c}_t\mathbf{1}_n^\top \\
&= \mathbf{R}_{t-1}\mathbf{U}_{\text{affine},t}\mathbf{U}_{\text{affine},t}^\top\mathbf{S}_{t-1}\mathbf{X}'_{\odot} + \mathbf{c}_t\mathbf{1}_n^\top \\
&= \mathbf{R}_{t-1}\mathbf{S}_{t-1}\mathbf{X}'_{\odot,t} + \mathbf{c}_t\mathbf{1}_n^\top.
\end{aligned}$$

Substituting the updated values of $\mathbf{X}'_{\odot,t}$ and $\mathbf{c}_t$ from Equations 9.14 and 9.15 and simplifying, using the fact that matrix multiplication is left- and right distributive, yields

$$\begin{aligned}
\therefore \mathbf{X}_t &= \mathbf{R}_{t-1}\mathbf{S}_{t-1}\mathbf{X}'_{\odot,t} + \mathbf{c}_t\mathbf{1}_n^\top \\
&= \mathbf{R}_{t-1}\mathbf{S}_{t-1}(\tilde{\mathbf{X}}'_{t-1} - \tilde{\mathbf{x}}'_{\min,t-1}\mathbf{1}_n^\top) + (\mathbf{c}_{t-1} + \mathbf{R}_{t-1}\mathbf{S}_{t-1}\tilde{\mathbf{x}}'_{\min,t-1})\mathbf{1}_n^\top \\
&= \mathbf{R}_{t-1}\mathbf{S}_{t-1}\tilde{\mathbf{X}}'_{t-1} + \mathbf{c}_{t-1}\mathbf{1}_n^\top - \mathbf{R}_{t-1}\mathbf{S}_{t-1}\tilde{\mathbf{x}}'_{\min,t-1}\mathbf{1}_n^\top + \mathbf{R}_{t-1}\mathbf{S}_{t-1}\tilde{\mathbf{x}}'_{\min,t-1}\mathbf{1}_n^\top \\
&= \mathbf{R}_{t-1}\mathbf{S}_{t-1}\tilde{\mathbf{X}}'_{t-1} + \mathbf{c}_{t-1}\mathbf{1}_n^\top \\
&= \tilde{\mathbf{X}}_{t-1},
\end{aligned}$$

which must be true, given that the set of untransformed input observations does not change between the end of the previous algorithm iteration and the calculation of the new transformation parameters of the next algorithm iteration.

Therefore, Equation 9.24 implies a mapping using the same original input points from Equation 9.23 expressed using the new, transformed input values $X'_t$ and transformation parameters $\mathbf{R}_t$, $\mathbf{S}_t$ and $\mathbf{c}_t$ such that the invariant properties (ii)–(iv) hold, concluding the proof. □

This iterative technique ensures that all of the invariant properties (i)–(iv) expected of $X'$

and $Y'$ are efficiently ensured at each algorithm iteration as the current set of observations $\mathcal{D}$ changes, as these changes may cause invariant properties to no longer hold for the transformations defined in the previous algorithm iteration. Using this approach avoids the need to explicitly store or manipulate the observed data in the original objective function space $X$ and $Y$. This is important, as this space can become ill-conditioned or face numerical issues, especially when data points are very close together, which can occur as the algorithm converges to a very small distance from the desired solution.

In summary, this cyclical process of recentring, rescaling and rotation ensures that the transformed $X'$ and $Y'$ remain well-conditioned at each algorithm iteration, even if the corresponding observations in the original objective function space $X$ and $Y$ become ill-conditioned or clustered closely together. The rotation performed in Equation 9.17 also ensures that the automatic relevance determination kernel, as defined in Equation 3.9, can effectively leverage local separability within the objective function during the rescaling in Equation 9.19, aligning the axes of each length-scale of the automatic relevance determination kernel with the axes of local separability, as seen in Figure 7.3.

## 9.2 Approximative Gaussian Process Hyperparameter Estimation

In the previous section that describes the transformation from the original observations $X$ and $Y$ to transformed representations $X'$ and $Y'$, the length-scale invariant property (iv) is preserved in Equation 9.19 by calculating the most likely length-scale hyperparameters $\hat{\boldsymbol{\ell}}$ for a GP fitted to the recentred and rotated observed inputs $X'_{\mathbb{C}}$ from Equation 9.17 and normalized output values $Y'$ from Equation 9.7. This fitted GP is also used to determine the next input point to sample from the objective function by maximizing the acquisition function based on this GP. Therefore, a set of optimal or near-optimal kernel hyperparameters need to be determined at each algorithm iteration to ensure efficient length-scale-based rescaling and a sufficiently accurate surrogate GP model of the objective function.

Instead of the conventional approach of a full re-estimation of the kernel hyperparameters $\hat{\boldsymbol{\theta}}$ using the MAP estimate of the marginal likelihood for a given set of hyperparameters from Equation 3.16 at each algorithm iteration (or once per some fixed number of algorithm iterations), an approximative scheme is adopted. In this approach, the approximate hyperparameters for the local GP model are calculated at each algorithm iteration using a small, fixed number of optimization steps, which are expected to tend towards the exact hyperparameters with subsequent *algorithm iterations*, not with additional *optimization steps per algorithm iteration*. Using a fixed number of optimization steps, instead of executing as many optimization steps as necessary for convergence of the hyperparameters, results in computational advantages by setting a fixed cap to the number of expensive operations with a complexity of $O(n^3)$ (recalculating the $\mathbf{K}^{-1}$ matrix from Section 3.1) performed during each algorithm iteration.

The rest of this section is structured as follows: Firstly, the choices and prior distributions of the hyperparameters are defined that influence the shape of the log marginal likelihood for a given set of hyperparameters, and, by extension, the Jacobian and Hessian matrices thereof. Next, the operations necessary to calculate the Jacobian and Hessian matrices are presented and several symmetries in these matrices are discussed that ease the computational cost of these calculations. Finally, the approximate maximization of the log marginal likelihood using

a single Newton or gradient ascent step, using the previously calculated Jacobian and Hessian matrices, combined with backtracking line search is given.

### 9.2.1 Hyperparameter Prior Distribution Selection

Recalling the definition of the squared exponential kernel function with automatic relevance determination from Section 3.2, the hyperparameters of this kernel consists of the signal and noise variances ($\sigma_f$ and $\sigma_n$) and length-scales for each dimension ($\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_d)$) in addition to the mean function $m(\cdot)$ of the GP. In the LABCAT algorithm, the length-scales are optimized using the log marginal likelihood from Equation 3.16 while the rest of these hyperparameters are calculated from the observed data directly at each algorithm iteration.

Firstly, the prior mean function $m(\cdot)$ of the GP is set to a constant to prevent excessive exploration, defined by the mean of the transformed output data $Y'$

$$m(\cdot) = \frac{1}{n}\sum_{i=1}^{n} y'_i, \tag{9.25}$$

and the noise variance in the chosen kernel function from Equation 3.9 is chosen to be set to a small value to function as a small "nugget" term for increased numerical stability [37], with the fixed[3] value of

$$\sigma_n = 10^{-6}. \tag{9.26}$$

Furthermore, while the standard approach is to optimize the value of the signal variance parameter $\sigma_f$ directly using the marginal likelihood, the value of this parameter is set to a fixed value of the standard deviation of $Y'$ (similar to the choice of prior distribution in the BADS [54] algorithm) at each algorithm iteration

$$\sigma_f = \mathrm{std}(Y'), \tag{9.27}$$

with the algorithm not very sensitive to this choice due to the continuous rescaling of the outputs described in Equation 9.7.

Next, the *logarithm* of the length-scales is optimized. The reasons for this choice is twofold: Firstly, this modification ensures that the length-scales remain strictly positive, as negative length-scales have no geometric interpretation and length-scales of zero would lead to a rank-deficient (and, therefore, singular) transformation in Equation 9.19. Secondly, since the inferred length-scales essentially function as trust-region expansion or contraction factors as mentioned in Section 8.1, the trust-region contraction would be very sensitive to length-scales smaller than one and vice versa. In other words, while inferred length-scales of, say, 0.1 and 10 are equivalent to contracting or expanding the trust region by a factor of 10, the distance from unity differs greatly. Finding the optimum length-scales with respect to the logarithm solves both of these problems, as the logarithm is strictly positive by definition and 0.1 and 10 are equidistant from 1 in logarithmic space.

It is also reasonable to assume, in the context of trust-region-based BO and the rescaling performed in the previous section, that the most likely length-scales for a GP fitted to a

[3]Since the values of $Y'$ are min-max normalized at each algorithm, the fixed value of $\sigma_n$ will always remain small relative to the observed outputs, even if the original observations $Y$ become very small during convergence.

trust region typically exhibit mostly gradual changes as the trust region shifts. Thus, if the most likely length-scales are calculated for a GP fitted to a locally constrained window and a new potential minimum is determined, causing a slight shift in the window's location, the hyperparameters for the new window are expected to be similar to the previous values. To incorporate this assumption, a Gaussian prior is placed over the logarithm of the length-scales (e.g., [66, 67, 90]). This augmentation effectively restrains the GP from making abrupt changes in hyperparameters, reinforcing the stability of the algorithm. Given the invariant property (iv), the value of the length-scale for a new algorithm iteration should be close to unity, therefore, the Gaussian prior is chosen to be centred on one. Therefore, the Gaussian prior (now centred on 0 in logarithmic space) is defined as

$$\ln \ell_i \sim \mathcal{N}(0, \sigma_{\boldsymbol{\ell}}^2) \quad \forall i \in \{1, \dots d\}. \tag{9.28}$$

For the standard deviation of this prior $\sigma_{\boldsymbol{\ell}}$, a default value of approximately 0.1 is suggested, such that, by the three-sigma rule of thumb,[4] the side lengths of the local trust region is unlikely to change by more than 30% per algorithm, as the length-scales can be expected to fall in the range $0.7 < \hat{\ell}_d < 1.3$. In the proposed, iterative hyperparameter estimation scheme of the LABCAT algorithm, this prior distribution provides both regularization and damping to the process of approaching the true hyperparameters with subsequent algorithm iterations by preventing absurd choices for the length-scales.

With these design choices, the hyperparameters to be optimized are reduced to the length-scales $\boldsymbol{\ell}$ of the squared exponential kernel with automatic relevance determination for the GP fitted in the transformed space of $X'_{\complement}$ and $Y'$. Using an adapted form of Equation 3.16, this problem is stated as

$$\hat{\boldsymbol{\ell}} = \underset{\boldsymbol{\ell}}{\operatorname{argmax}} (\log p(Y' \mid X'_{\complement}, \boldsymbol{\theta}) + \log p(\ln \boldsymbol{\ell} \mid \sigma_{\boldsymbol{\ell}})) \tag{9.29}$$

with the prior distribution term now being log-normally distributed

$$\log p(\ln \boldsymbol{\ell} \mid \sigma_{\boldsymbol{\ell}}) = \log\big(\mathcal{N}(\mathbf{0}, \sigma_{\boldsymbol{\ell}}^2 \mathbf{I})\big). \tag{9.30}$$

The derivatives of the log marginal likelihood surface from Equation 3.13 (without a prior distribution for $\boldsymbol{\theta}$), with respect to the logarithm of the length-scales $\boldsymbol{\ell}$ [91], are given by the Jacobian defined as

$$\mathbf{J}_{\boldsymbol{\ell}} = \nabla_{\ln \boldsymbol{\ell}} \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta}) = \left[ \frac{\partial \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta})}{\partial \ln \ell_i} \right]_{1 \le i \le d} \in \mathbb{R}^{d \times 1} \tag{9.31}$$

and Hessian

$$\mathbf{H}_{\boldsymbol{\ell}} = \nabla^2_{\ln \boldsymbol{\ell}} \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta}) = \left[ \frac{\partial^2 \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta})}{\partial \ln \ell_i \, \partial \ln \ell_j} \right]_{1 \le i,j \le d} \in \mathbb{R}^{d \times d}, \tag{9.32}$$

[4] A value sampled three standard deviations from the mean of a Gaussian distribution would be a part of the tail of the distribution and, as a result, be quite small. As the Gaussian prior is multiplied with the marginal likelihood (both strictly positive), the resulting marginal likelihood three standard deviations away from the mean of the Gaussian prior would also be quite small and very unlikely to be the MAP estimate of the resulting posterior likelihood.

where the partial derivatives $\frac{\partial \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta})}{\partial \ln \ell_i}$ and $\frac{\partial^2 \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta})}{\partial \ln \ell_i \, \partial \ln \ell_j}$ for the squared exponential kernel with automatic relevance determination from Equation 3.9 are given in the next section.

Incorporating the prior distribution over the length-scales from Equation 9.30, the log marginal likelihood maximized in Equation 9.29 (derived from Equations 3.13 and 3.16) is augmented with this prior distribution (using the logarithm of the multivariate Gaussian and ignoring normalization constants) and is given by

$$\begin{aligned} \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta}; \sigma_{\boldsymbol{\ell}}) &= \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta}) + \log p(\ln \boldsymbol{\ell} \mid \sigma_{\boldsymbol{\ell}}) \\ &= \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta}) - \frac{(\ln \boldsymbol{\ell}^{\top})(\ln \boldsymbol{\ell})}{2\sigma_{\boldsymbol{\ell}}^2}, \end{aligned} \tag{9.33}$$

with the Jacobian defined in Equation 9.31 augmented as

$$\mathbf{J}_{\boldsymbol{\ell} \mid \sigma_{\boldsymbol{\ell}}} = \nabla_{\ln \boldsymbol{\ell}} \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta}; \sigma_{\boldsymbol{\ell}}) = \mathbf{J}_{\boldsymbol{\ell}} - \frac{1}{\sigma_{\boldsymbol{\ell}}^2} \ln \boldsymbol{\ell} \tag{9.34}$$

and the Hessian defined in Equation 9.32 augmented as

$$\mathbf{H}_{\boldsymbol{\ell} \mid \sigma_{\boldsymbol{\ell}}} = \nabla^2_{\ln \boldsymbol{\ell}} \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta}; \sigma_{\boldsymbol{\ell}}) = \mathbf{H}_{\boldsymbol{\ell}} - \frac{1}{\sigma_{\boldsymbol{\ell}}^2} \mathbf{I}. \tag{9.35}$$

### 9.2.2 Jacobian and Hessian Matrix Calculation

During the optimization of the length-scales in the previous subsection, first and second derivatives of the log marginal likelihood for a GP constructed using the observations $X'_{\complement}$ and $Y'$ (derived from Equation 3.13), with respect to the logarithm of the length-scales from kernel function defined in Equation 3.9, are calculated. This subsection provides the definitions of the partial derivatives that define the entries of these Jacobian $\mathbf{J}_{\boldsymbol{\ell}}$ and Hessian $\mathbf{H}_{\boldsymbol{\ell}}$ matrices, as well as present several symmetries in these partial derivatives that reduce the memory footprints and computational overhead needed for these calculations.

Using the results as derived by Moore et al. [91], Zhang and Leithead [92], and Rasmussen [30, Ch. 5],[5] the partial derivatives that define the entries of the matrices in Equations 9.31 and 9.32 are given for the Jacobian by

$$\begin{aligned} \frac{\partial \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta})}{\partial \ln \ell_i} &= \frac{1}{2} (\mathbf{y}' - \mathbf{m}')^{\top} \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \ln \ell_i} \mathbf{K}^{-1} (\mathbf{y}' - \mathbf{m}') \\ &\quad - \frac{1}{2} \operatorname{tr}\left( \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \ln \ell_i} \right) \ \forall i \in \{1, \ldots, d\}, \end{aligned} \tag{9.36}$$

and for the Hessian by

[5] As stated by Zhang and Leithead [92] and Rasmussen [30, Ch. 5], directly calculating the matrix-matrix products in Equations 9.31 and 9.32 should be avoided as far as possible. Instead, the products $\mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \ell_i}$ should be cached, matrix-vector products should be prioritized and only the products needed for the trace terms should be calculated.

$$\frac{\partial^2 \log p(Y' \mid X'_{\complement}, \boldsymbol{\theta})}{\partial \ln \ell_i \, \partial \ln \ell_j} = \frac{1}{2} \operatorname{tr}\left( \mathbf{K}^{-1} \frac{\partial^2 \mathbf{K}}{\partial \ln \ell_i \, \partial \ln \ell_j} \right) - \frac{1}{2} \operatorname{tr}\left( \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \ln \ell_j} \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \ln \ell_i} \right) + (\mathbf{y}' - \mathbf{m}')^{\top} \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \ln \ell_j} \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \ln \ell_i} \mathbf{K}^{-1} (\mathbf{y}' - \mathbf{m}') - \frac{1}{2} (\mathbf{y}' - \mathbf{m}')^{\top} \mathbf{K}^{-1} \frac{\partial^2 \mathbf{K}}{\partial \ln \ell_i \, \partial \ln \ell_j} \mathbf{K}^{-1} (\mathbf{y}' - \mathbf{m}') \;\; \forall i, j \in \{1, \ldots, d\}, \tag{9.37}$$

where the vectors $\mathbf{y}'$ and $\mathbf{m}'$ are constructed similarly to Equations 3.4 and 3.5, respectively, using the transformed observed outputs $Y'$ with the chosen, constant mean function defined in Equation 9.25 of the previous section.

The derivative factors in Equations 9.36 and 9.37 are specifically calculated with respect to the logarithm of the length-scales by transforming the derivatives of the kernel matrix $\mathbf{K}$ from Equation 3.3 according to

$$\frac{\partial \mathbf{K}}{\partial \ln \ell_i} = \frac{\partial \mathbf{K}}{\partial \ell_i} \frac{\partial \ell_i}{\partial \ln \ell_i}, \tag{9.38}$$

accomplished by multiplying the derivative of the length-scale with respect to its logarithm, derived using the dummy variable $v_i$ as

$$\begin{aligned} v_i = \ln \ell_i \implies \ell_i = e^{v_i} \\ \therefore \frac{\partial \ell_i}{\partial \ln \ell_i} = \frac{\partial e^{v_i}}{\partial v_i} = e^{v_i} = \ell_i. \end{aligned} \tag{9.39}$$

The kernel matrix derivative, with respect to the logarithm of the length-scales, now becomes

$$\frac{\partial \mathbf{K}}{\partial \ln \ell_i} = \begin{bmatrix} \frac{\partial k(\mathbf{x}_1, \mathbf{x}_1)}{\partial \ln \ell_i} & \frac{\partial k(\mathbf{x}_1, \mathbf{x}_2)}{\partial \ln \ell_i} & \cdots & \frac{\partial k(\mathbf{x}_1, \mathbf{x}_n)}{\partial \ln \ell_i} \\ \frac{\partial k(\mathbf{x}_2, \mathbf{x}_1)}{\partial \ln \ell_i} & \frac{\partial k(\mathbf{x}_2, \mathbf{x}_1)}{\partial \ln \ell_i} & \cdots & \frac{\partial k(\mathbf{x}_2, \mathbf{x}_n)}{\partial \ln \ell_i} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial k(\mathbf{x}_n, \mathbf{x}_1)}{\partial \ln \ell_i} & \frac{\partial k(\mathbf{x}_n, \mathbf{x}_2)}{\partial \ln \ell_i} & \cdots & \frac{\partial k(\mathbf{x}_n, \mathbf{x}_n)}{\partial \ln \ell_i} \end{bmatrix}. \tag{9.40}$$

Note that, since the SE kernel is symmetric (i.e., $k(\mathbf{x}_p, \mathbf{x}_q) = k(\mathbf{x}_q, \mathbf{x}_p)$), the derivative of the kernel matrix is also symmetric

$$\frac{\partial \mathbf{K}}{\partial \ln \ell_i} = \left( \frac{\partial \mathbf{K}}{\partial \ln \ell_i} \right)^{\top}, \tag{9.41}$$

allowing the kernel matrix derivatives to be fully described by only calculating the upper or lower triangular portions of the matrix.

Calculating the entries of Equation 9.40 simply involve taking the derivative of Equation 3.9 with respect to $\ell_i$

$$\frac{\partial k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ell_i} = k(\mathbf{x}_p, \mathbf{x}_q) \cdot \frac{(x_{p_i} - x_{q_i})^2}{\ell_i^3} \tag{9.42}$$

and multiplying by $\ell_i$ to obtain the derivative with respect to the logarithm of $\ell_i$

$$\frac{\partial k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ln \ell_i} = \frac{\partial k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ell_i} \frac{\partial \ell_i}{\partial \ln \ell_i} = k(\mathbf{x}_p, \mathbf{x}_q) \cdot \frac{(x_{p_i} - x_{q_i})^2}{\ell_i^2}. \tag{9.43}$$

Using these results, the partial second derivatives of the kernel matrix, with respect to the length-scales $\frac{\partial^2 \mathbf{K}}{\partial \ln \ell_i \, \partial \ln \ell_j}$, can be determined. The entries of these matrices, for the case that $i \neq j$, are given as

$$\frac{\partial^2 k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ln \ell_j \, \partial \ln \ell_i} = \frac{\partial k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ln \ell_j} \cdot \frac{(x_{p_i} - x_{q_i})^2}{\ell_i^2}. \tag{9.44}$$

Expanding and rearranging terms in this formula yields

$$\begin{aligned} \frac{\partial^2 k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ln \ell_j \, \partial \ln \ell_i} &= k(\mathbf{x}_p, \mathbf{x}_q) \cdot \frac{(x_{p_j} - x_{q_j})^2}{\ell_j^2} \cdot \frac{(x_{p_i} - x_{q_i})^2}{\ell_i^2} \\ &= k(\mathbf{x}_p, \mathbf{x}_q) \cdot \frac{(x_{p_i} - x_{q_i})^2}{\ell_i^2} \cdot \frac{(x_{p_j} - x_{q_j})^2}{\ell_j^2} \\ &= \frac{\partial k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ln \ell_i} \cdot \frac{(x_{p_j} - x_{q_j})^2}{\ell_j^2} \\ &= \frac{\partial^2 k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ln \ell_i \, \partial \ln \ell_j}, \end{aligned} \tag{9.45}$$

implying the following symmetry in the log marginal likelihood Hessian matrix $\mathbf{H}_{\boldsymbol{\ell}}$

$$\frac{\partial^2 \mathbf{K}}{\partial \ln \ell_i \, \partial \ln \ell_j} = \left( \frac{\partial^2 \mathbf{K}}{\partial \ln \ell_i \, \partial \ln \ell_j} \right)^\top. \tag{9.46}$$

Lastly, for the case where $i = j$ (the main diagonal of $\mathbf{H}_{\boldsymbol{\ell}}$ from Equation 9.32), the derivative is determined to be

$$\frac{\partial^2 k(\mathbf{x}_p, \mathbf{x}_q)}{\partial^2 \ln \ell_i} = \frac{\partial k(\mathbf{x}_p, \mathbf{x}_q)}{\partial \ln \ell_i} \left( \frac{(x_{p_i} - x_{q_i})^2}{\ell_i^2} - 2 \right). \tag{9.47}$$

The identified symmetries in the kernel matrix partial derivatives reduce the computational overhead needed to calculate as well as the memory footprints of the Jacobian and Hessian from Equations 9.36 and 9.37, which are of the order $O(n^2 d)$ and $O(n^3 d)$ for general set of $d$ hyperparameters, respectively [91, 92].

### 9.2.3 Marginal Likelihood Maximization

After the calculation of the Jacobian and Hessian matrices $\mathbf{H}_{\boldsymbol{\ell} \,|\, \sigma_{\boldsymbol{\ell}}}$ and $\mathbf{J}_{\boldsymbol{\ell} \,|\, \sigma_{\boldsymbol{\ell}}}$ from Equations 9.34 and 9.35 of the log marginal likelihood defined in Equation 9.33, it is now possible to partially maximize this marginal likelihood defined by the calculated matrices to determine the approximate MAP estimate for the most likely length-scales $\hat{\boldsymbol{\ell}}$. This maximization is performed using a single Newton or gradient step (based on the local geometry of the log

marginal likelihood) per algorithm iteration from the centre of the multivariate Gaussian prior distribution for the length-scales from Equation 9.30.

If $\mathbf{H}_{\boldsymbol{\ell}\,|\,\sigma_{\boldsymbol{\ell}}}$ is negative definite,[6] it can be concluded that the current hyperparameters are in a concave region of the likelihood space. Consequently, a single second-order Newton step from the starting point of the centre of the chosen multivariate normal prior distribution (or, a vector of zeroes $\mathbf{0}$ for the starting log length-scales) from Equation 9.30, with step size factor $\gamma$, can be effectively employed. This second-order Newton step is given by

$$\ln \hat{\boldsymbol{\ell}} = \mathbf{0} - \gamma \mathbf{H}^{-1}_{\boldsymbol{\ell}\,|\,\sigma_{\boldsymbol{\ell}}} \mathbf{J}_{\boldsymbol{\ell}\,|\,\sigma_{\boldsymbol{\ell}}}. \tag{9.48}$$

If $\mathbf{H}_{\boldsymbol{\ell}\,|\,\sigma_{\boldsymbol{\ell}}}$ is not negative definite (in effect, the current hyperparameters are not in a concave region of the log marginal likelihood space) a simple gradient ascent step is used, given by

$$\ln \hat{\boldsymbol{\ell}} = \mathbf{0} + \gamma \mathbf{J}_{\boldsymbol{\ell}\,|\,\sigma_{\boldsymbol{\ell}}}. \tag{9.49}$$

For both of the second-order Newton- and gradient descent step, the step size factor $\gamma$ is selected using a backtracking line search [95] defined by the recurrence relation

$$\gamma_i = \tau \gamma_{i-1}, \quad i \in \{2, \ldots, 5\},$$

with the initial value

$$\gamma_1 = 1. \tag{9.50}$$

In other words, the step size $\gamma$ is reduced from 1 by a factor of $\tau$ until the marginal likelihood improves compared to the marginal likelihood of using length-scales of all unity or for a maximum of 5 iterations, a relaxed stopping condition compared to the more stringent sufficient decrease criterion of the Armijo-Goldstein condition [95]. Similarly to values suggested by Armijo, the value of $\tau = \frac{1}{2}$ is set for the second-order Newton step in Equation 9.48 and is set to $\tau = \frac{1}{10}$ for the gradient descent in Equation 9.49, due to the lack of preconditioning of the gradient step. Formally, this backtracking line search step to refine the approximation to the MAP estimate of the length-scales $\hat{\boldsymbol{\ell}}$ can be given as

$$\hat{\boldsymbol{\ell}} = \min\{i \in \{1, \ldots, 5\} \,|\, \mathcal{L}_{\boldsymbol{\theta}}(\hat{\boldsymbol{\ell}}_i) \geq \mathcal{L}_{\boldsymbol{\theta}}(\mathbf{1}_d)\}, \tag{9.51}$$

where the symbol $\hat{\boldsymbol{\ell}}_i$ denotes the length-scales obtained from using a step size of $\gamma_i$ for the Newton or gradient step from Equations 9.48 and 9.49 and $\mathcal{L}_{\boldsymbol{\theta}}$ is a function that evaluates the log marginal likelihood from Equation 9.33 at the specified length-scales $\hat{\boldsymbol{\ell}}_i$ or using unit length-scales.

Considering the rescaling of the input data performed in Equation 9.19, the calculation of the most likely length-scales during each algorithm iteration can be interpreted as an indication

[6] The naïve method to determine negative definiteness would be to check if all eigenvalues of $\mathbf{H}_{\boldsymbol{\ell}\,|\,\sigma_{\boldsymbol{\ell}}}$ are negative after an eigendecomposition from Equation 5.6 with a complexity of $O(d^3)$. A more efficient method would be to check if the matrix can be successfully decomposed using a Cholesky decomposition [62] or if every Gershgorin disc [93] of the matrix is strictly negative (similarly, checking the minimum diagonal dominance using the modified Gershgorin method of Deville [94] with better bounds for real, symmetric matrices), with these methods of the order $O(\frac{1}{3}d^3)$ and $O(d^2)$, respectively. In most practical applications of the LABCAT algorithm, however, the number of observations are much larger than the objective function dimension ($n \gg d$) and the operations of the order $O(n^3)$ should dominate.

to expand or contract each of the dimensions of the transformed input data $X'$. If $X'$ is never recentred, it is also reasonable to expect that additional observations could allow the GP to construct a more accurate model of the objective function and the length-scales of this better GP model would tend to unity.

## 9.3 Fixed Trust Region Definition

As previously stated in Section 4.2, a key mechanism of trust-region-based BO is limiting the region of the acquisition function around the best candidate solution in which the next point is observed by means of a trust region. In effect, finding this constrained maximum of the acquisition function determines the next point to be observed from the objective function and, therefore, the trust region is a key mechanism in the creation of the local GP surrogate model. In the LABCAT algorithm, after the set of observations $X$ is transformed to $X'$ according to the invariant properties (i)–(iv) — specifically noting invariant property (ii) and (iv) which state that the observed inputs are centred on the current minimum candidate ($\mathbf{x}'_{\text{min}} = \begin{bmatrix}0 & \dots & 0\end{bmatrix}^\top \in X'$) and the most likely length-scales of the kernel are rescaled to unity ($\hat{\boldsymbol{\ell}} \mapsto (1, 1, \dots, 1)$) — a trust region $\Omega'_{\text{TR}}$ is constructed in the space of $X'$ as a closed, compact $d$-cube with a side length of $2\beta$ and the constrained maximum of the acquisition function is found using this trust region. Using the Cartesian product, this fixed trust region is defined as the hypercube

$$\Omega'_{\text{TR}} = [-\beta, \beta]^d, \tag{9.52}$$

where $\beta$ is a tunable parameter that captures the trade-off between the exploration of the objective function and the exploitation of the region surrounding the current minimum candidate solution. Small values for $\beta$ strongly encourage local exploitation of the minimum candidate solution, but may lead to small step sizes between successive candidate solutions. In the case where $\beta$ tends to infinity, the algorithm will search for the next point in an unconstrained manner and revert to the global optimization of standard BO. For this parameter, values in the interval $0.1 \leq \beta \leq 1$ are recommended according to the rough heuristic $\beta \approx \frac{1}{d}$. From practical experience, this range of values provides a good balance between exploration of the objective function and encouraging convergence to a local optimum.

It is important to reiterate the difference between the trust region used by LABCAT and those used by other, more classical methods such as those described in Section 2.5 and 4.2. Instead of the size of the trust region being directly modified inside the fixed space of the original observations $X$, in the LABCAT algorithm the size of the trust region is fixed in the transformed space of observations $X'$ and it is this transformed space that is scaled, rotated and translated. In effect, this induces a trust region in the original objective function space of $X$ without requiring the transformation of $X'$ back to $X$.

The acquisition function bounded by the trust region, chosen as the expected improvement function (Section 4.1),[7] is maximized using $10d$ random samples uniformly distributed across the fixed trust region $\Omega'_{\text{TR}}$. This technique is similar to the one used by the TRLBO algorithm [55] and with the similar justification that the expected acquisition function optimization

[7] Note that many alternative acquisition functions have been proposed [69, 70, 96] and while we have chosen expected improvement for simplicity and due to popularity, other acquisition functions could potentially be substituted for expected improvement in the proposed algorithm.

error from using random search (compared to the standard approach of a more intensive multi-start optimization procedure) will become smaller and smaller as the trust region shrinks. The results obtained from this maximization are also validated against the original constraints of the objective function $\Omega$ using rejection sampling after transforming the points back into the original objective function space using the transform defined in Equation 9.1. In other words, the point $\mathbf{x}'_\alpha$ is found, similarly to Equation 4.5, by maximizing the expected improvement over the intersection of $\Omega$ and $\Omega'_{\text{TR}}$,[8] or

$$\mathbf{x}'_\alpha = \underset{\substack{\mathbf{x}'_* \in \Omega'_{\text{TR}} \\ \mathbf{x}_* \in \Omega}}{\operatorname{argmax}} \alpha_{\text{EI}}(\mathbf{x}'_*;\, (X', Y')). \tag{9.53}$$

To sample the objective function using this input point, calculated in the transformed space, the point is transformed to $\mathbf{x}_\alpha$ in the original objective function space using the transform defined in Equation 9.1. This input point is then sampled from the objective function ($f(\mathbf{x}_\alpha) = y_\alpha$) and, after transforming the observed output value to $y_\alpha$ using Equation 9.3, added to the set of observed data.

## 9.4 Observation Discarding Strategy

Apart from limiting the region from which the next observation should be chosen, the trust region $\Omega'_{\text{TR}}$ is also used to determine which observed points from $X'$ and $Y'$ should be preserved at each algorithm iteration. Observations outside this trust region are assumed to no longer contribute significant information and, therefore, are discarded. This keeps the number of observations in the GP surrogate model and, by extension, the computation time per algorithm iteration relatively constant across the runtime of the algorithm, alleviating the noted computational slowdown of standard BO (Section 3.1).[9]

A minimum number of observations are preserved at each algorithm iteration, even if some fall outside the trust region, as there may be cases where discarding too many observations may cause the observation set to become rank-deficient, an occurrence that may lead the GP to make explosive changes to the most likely length-scales. This cache size factor is denoted by $\rho$, such that the minimum number of preserved observations is the cache size factor $\rho$ multiplied by the objective function dimensionality $d$. The operation applied to $X'$, removing the observations in the set $X'_{\notin \Omega'_{\text{TR}}}$ if the size of $X'$ is larger than $\rho d$, is defined as

$$T_{\notin \Omega'_{\text{TR}}}(X', Y') \coloneqq \begin{cases} (X', Y') & |X'| \leq \rho d \\ (X' \setminus X'_{\notin \Omega'_{\text{TR}}},\ Y' \setminus \{y'_i \,|\, \mathbf{x}'_i \in X'_{\notin \Omega'_{\text{TR}}}\}) & |X'| > \rho d. \end{cases} \tag{9.54}$$

[8] While not explicitly handled as part of the expected improvement maximization, nonlinear or non-bound constraints can theoretically be incorporated during the rejection sampling or by adding a barrier function to the objective function.

[9] It is also not necessary to recalculate the kernel matrix of the GP, and the inverse or Cholesky decomposition thereof, for the reduced set of observations. As noted in Section 4.1, for a GP with fixed hyperparameters (as in this case before maximizing the acquisition function), the corresponding rows and columns of the elements in $X'_{\notin \Omega'_{\text{TR}}}$ can be removed from $\mathbf{K}$ while $\mathbf{K}^{-1}$ can be updated efficiently in terms of the Schur complement [72] or rank-1 downdates of the Cholesky decomposition of $\mathbf{K}$, both of the order $O(n^2)$ [27, Ch. 9] for each removed observation.

The set of input observations to be discarded $X'_{\notin\Omega'_{\mathrm{TR}}}$ are chosen, prioritizing older observations, to be those that fall outside the current trust region ($\forall \mathbf{x}' \in X'_{\notin\Omega'_{\mathrm{TR}}}$, $\mathbf{x}' \notin \Omega'_{\mathrm{TR}}$) until the size of $X'$ reaches the $\rho d$ threshold.[10] The corresponding elements from $Y'$ are also removed to ensure that $X'$ and $Y'$ retain a one-to-one correspondence. Note that in this operation, the current minimum candidate solution $\mathbf{x}'_{\mathrm{min}}$ is guaranteed to be preserved due to invariant property (ii), which guarantees that this candidate is moved to the origin, an element of $\Omega'_{\mathrm{TR}}$ by definition.

The cache size factor $\rho$ is a user-specified parameter and a poor choice thereof may lead to suboptimal performance. For instance, if $\rho$ is set too low, the model GP constructed using $X'$ and $Y'$ may have too few observations to model the objective function adequately. Conversely, if $\rho$ is set too high, the algorithm may become sluggish as it struggles to discard old, non-informative observations (that slow down the $O(n^3)$ matrix operations and possibly misinform the local GP surrogate model) quickly enough to keep up with the moving trust region. Bearing these remarks in mind, a value in the interval $5 \leq \rho \leq 10$ is recommended.

## 9.5 Algorithm Initialization and Termination

During the initialization of the LABCAT algorithm, similarly to standard BO discussed in Section 4.1, a set of initial points are chosen to be evaluated before the main loop begins, with this selection strategy known as the design of experiment (DoE). Using the provided input domain $\Omega$, the DoE for the initial GP surrogate model $X_0$ is distributed according to a Latin hypercube design [73] with $2d + 1$ points to ensure full rank. Latin hypercube sampling is a popular choice as the DoE for trust-region-based BO methods [50–52, 55] to avoid clustering of the initial points, which might occur with random sampling, and to capture as much projected coverage along the objective function's coordinate axes as possible.

After observing these initial input points $X_0$ from the objective function to obtain the initial observed outputs $Y_0$, the upper and lower bounds placed on the objective function used to define $\Omega$ in Equation 4.1 are used to initialize the respective transformed representations $\tilde{X}'_0$ and $\tilde{Y}'_0$. To determine the initialized transformed representations, we adapt the mechanisms used to enforce the invariant properties (i), (ii) and (iv) for this initialization. The modifications of this adapted mechanism are that $\tilde{X}'_0$ is centred on the midpoint of the bounds, no rotation is performed, and the bounds are rescaled to lie on the hypercube $[-1, 1]^d$ (in other words, unit length from the origin in each dimension), similarly to the domain rescaling performed by several other trust-region-based BO methods [51, 52, 54]. These modifications ensure that initial observations from the DoE are well-conditioned, but it should also be noted that the invariant properties (i)–(iv) temporarily do not hold until the first algorithm iteration completes. The modified transformation to calculate the initial $\tilde{X}'_0$ and $\tilde{Y}'_0$ is given by

$$(\tilde{X}'_0,\, \tilde{Y}'_0) = T_\Omega(X_0, Y_0, \Omega) \coloneqq \left\{ \left( \mathbf{S}_0^{-1}(\mathbf{x}_{i,\,0} - \mathbf{c}_0), \frac{y_{i,\,0} - y_{\min,\,0}}{y_{\max,\,0} - y_{\min,\,0}} \right) \,\middle|\, i \in \{1, \ldots, n\} \right\} \tag{9.55}$$

In addition to the initial values of the transformed inputs and outputs, the input transformation parameters $\mathbf{S}_0$, $\mathbf{R}_0$ and $\mathbf{c}_0$ from Equation 9.1 are initialized using $\Omega$ according to

[10] It was also considered to score the observed points by their relative importance using the compression metric based on Bayesian quadrature developed by Visser et al. [97]. However, it was found that the algorithm would become too hesitant to discard old observations that still have a large influence on the shape of the surrogate model and most likely length-scales, inhibiting performance and preventing convergence.

$$\mathbf{S}_0^{-1} = \operatorname{diag}\left(\frac{|\Omega_1^{\max} - \Omega_1^{\min}|}{2}, \ldots, \frac{|\Omega_d^{\max} - \Omega_d^{\min}|}{2}\right)^{-1},$$
$$\mathbf{R}_0 = \mathbf{I},$$
$$\text{and} \quad \mathbf{c}_0 = \left[\frac{\Omega_1^{\max} + \Omega_1^{\min}}{2} \quad \ldots \quad \frac{\Omega_d^{\max} + \Omega_d^{\min}}{2}\right]^\top \tag{9.56}$$

with the output transformation parameters $a_0$ and $b_0$ from Equation 9.3 initialized as

$$a_0 = y_{\max,0} - y_{\min,0} \quad \text{and} \quad b_0 = y_{\min,0}. \tag{9.57}$$

As previously discussed with standard BO (Section 4.1), the LABCAT algorithm also has no specific convergence criterion and may terminate as specified by the user if either (i) the current candidate minimum observed output $y_{\min}$ is less than some target value, (ii) if the range of output values in $Y$ (captured by the variable $a$) falls below some tolerance or (iii) the maximum objective function evaluation budget is reached.

## 9.6 Algorithm Pseudocode and Discussion

Synthesizing the detailed descriptions given in this chapter of the constituent components of the LABCAT algorithm, as seen in Figure 6.1, a description of the LABCAT algorithm is given in Algorithm 4.

The modified trust-region-based BO loop from Algorithm 3 is visible (lines $5-17$), with the additions of the transformation of the observed data (lines 3, $6-8$ and 11), determining the optimal length-scales (line 10) and the discarding of observations (line 12).

The mechanisms though which the objectives of this dissertation (from Section 1.2) are addressed can now be identified, with these objectives being to develop a trust-region-based BO method that (i) is resistant to computational slowdown, (ii) is adaptable to non-stationary and ill-conditioned functions without kernel engineering, and (iii) exhibits good convergence characteristics. Firstly, using the greedy data discarding strategy defined in Section 9.3 (line 12), the number of observations used to construct the local GP surrogate is kept relatively constant at each algorithm iteration. This avoids the computational slowdown of standard BO with more observations noted in Section 3.1. Secondly, the use of a local trust region based on the local length-scales of GP surrogate (line 13) allows the algorithm to adapt to the local shape of a non-stationary objective function. The rotation of the trust region using the weighted principal components (line 8) also allows the trust region to adapt to ill-conditioning or separability of the objective function in arbitrary directions. Lastly, the use of a transformed representation of the observed data $X'$ and $Y'$ that is forced to be well-conditioned allows the LABCAT algorithm to converge much closer to a solution before encountering the numerical issues found in standard BO.

**Algorithm 4** LABCAT

**Input:** Objective function $f$, Bounds $\Omega$, Trust region size factor $\beta$, Observation cache size factor $\rho$, Length-scale prior standard deviation $\sigma_{\boldsymbol{\ell}}$

Select initial input points $X_0$ using Latin hypercube sampling over $\Omega$
$Y_0 \leftarrow \{f(\mathbf{x}) \,|\, \mathbf{x} \in X_0\}$ ▷ Evaluate objective function at initial input points
$(\tilde{X}'_0,\, \tilde{Y}'_0) \leftarrow T_\Omega(X_0, Y_0, \Omega)$ ▷ Initialize transformed representation of observed data (see Eq. 9.55)
$\Omega'_{\mathrm{TR}} \leftarrow [-\beta, \beta]^d$ ▷ Construct fixed trust region hypercube (see Sec. 9.3)
**while not** converged **do**
    $Y'_t \leftarrow T_{\updownarrow, t}(\tilde{Y}'_{t-1})$ ▷ Normalize observed output values (see Eq. 9.7)
    $X'_{\odot, t} \leftarrow T_{\odot, t}(\tilde{X}'_{t-1})$ ▷ Centre observed inputs using current minimum candidate (see Eq. 9.14)
    $X'_{\complement, t} \leftarrow T_{\complement, t}(X'_{\odot, t})$ ▷ Rotate observed inputs using weighted principal components (see Eq. 9.17)
    $\mathcal{GP}(m(\cdot), k_{\mathrm{SE}}(\cdot,\cdot); (X'_{\complement, t}, Y'_t))$ ▷ Construct GP with an SE kernel with ARD to $X'_{\complement}$ and $Y'$ (see Ch. 3)
    $\hat{\boldsymbol{\ell}} \leftarrow \operatorname{argmax}_{\boldsymbol{\ell}}(\log p(Y'_t \,|\, X'_{\complement, t}, \boldsymbol{\ell}) + \log p(\boldsymbol{\ell} \,|\, \sigma_{\boldsymbol{\ell}}))$ ▷ Find most likely length-scales (see Sec. 9.2)
    $X'_t \leftarrow T_{\leftrightarrow, t}(X'_{\complement, t})$ ▷ Rescale obs. inputs with most likely length-scales (see Eq. 9.19)
    $(X'_t, Y'_t) \leftarrow T_{\notin \Omega'_{\mathrm{TR}}, t}(X'_t, Y'_t)$ ▷ Discard observations if over $\rho d$ threshold (see Eq. 9.54)
    $\mathbf{x}'_{\alpha, t} \leftarrow \operatorname{argmax}_{\substack{\mathbf{x}'_* \in \Omega'_{\mathrm{TR}} \\ \mathbf{x}_* \in \Omega}} \alpha_{\mathrm{EI}}(\mathbf{x}'_*;\, (X'_t, Y'_t))$ ▷ Maximize EI acquisition function over trust region and bounds using $10d$ random samples (see Eq. 4.7)
    $y_{\alpha, t} \leftarrow f(\mathbf{x}_{\alpha, t})$ ▷ Evaluate selected input point from objective function, transformed according to Eq. 9.1
    $(\tilde{X}'_t, \tilde{Y}'_t) \leftarrow \{(\mathbf{x}'_\alpha, y'_\alpha)\} \cup (X'_t, Y'_t)$ ▷ Append observation, with output value transformed according to Eq. 9.3
**end while**
**return** $(\mathbf{x}_{\min},\, y_{\min})$ ▷ Return minimum observed candidate solution

## 9.7 Computational Complexity

To analyze computational complexity of the LABCAT algorithm, the operations that predominantly influence the asymptotic complexity of a single algorithm iteration are listed with the respective complexities in Table 9.1 in terms of the current number of observations $n$ and dimensionality of the objective function $d$.

| Operation | Example equation | Computational complexity |
|---|---|---|
| Marginal likelihood Hessian | (9.32) | $O(n^3d)$ |
| Kernel matrix inversion | (3.3) | $O(n^3)$ |
| Marginal likelihood Jacobian | (9.31) | $O(n^2d)$ |
| SVD | (9.16) | $O(n^2d)$ |
| Rotation matrix multiplication | (9.17) | $O(nd^2)$ |
| Scaling matrix multiplication | (9.19) | $O(nd)$ |
| Matrix-vector addition/subtraction | (9.55) | $O(nd)$ |

Table 9.1: Computational complexity of operations that predominantly influence the complexity of the LABCAT algorithm with example equations where the respective operations are applied. Operations are ordered by noting that, in almost all cases, $n \gg d$.

From this table, it can be concluded that the asymptotic complexity of the LABCAT algorithm, performed for a total of $N$ iterations, to be $O(Nn^3d)$. At each algorithm iteration, due to the user-specified observation cache factor $\rho$ limiting the number of stored observations, the number of observations at each algorithm iteration is expected to remain relatively constant $n \approx \rho d$. By substituting this value of $n$ in $O(Nn^3d)$, the complexity of a single instance of the LABCAT algorithm would, therefore, be approximately $O(N\rho^3d^4)$ which can be further simplified to $O(N)$ (in effect, linear in terms of the number of iterations), since the values of $\rho$ and $d$ are fixed at the start of the algorithm instance.

In this chapter, the proposed LABCAT algorithm has been presented as a synthesis of the length-scale-based rescaling of Chapter 8 and the weighted-principal-component-based rotation of Chapter 7 with the trust-region-based BO framework reviewed in Chapter 4. A detailed description of the algorithm as well as the mechanisms used to achieve the aims of this dissertation have been outlined and it is this form of the algorithm that is used for a comparison with other state-of-the-art optimization algorithms, performed in the next chapter.

# Chapter 10

# Experimental Results

This chapter presents the results of a numerical performance analysis of the proposed LABCAT algorithm, which is based on two computational experiments. The first experiment, discussed in Section 10.1, compares the LABCAT algorithm with existing trust-region-based BO algorithms by applying them to several well-known, synthetic optimization test functions selected to encompass a range of objective function characteristics. These test functions are used to analyze the rate and limit of convergence as well as the computational slowdown of the tested algorithms, both known and identified shortcomings of standard BO. The second experiment applies the LABCAT algorithm and trust-region-based BO algorithms, as well as algorithms from the wider field of derivative-free optimization, to the BBOB test suite using the comparing continuous optimizers (COCO) benchmarking software [40], presented in Section 10.2. This extensive test suite is designed to be a representative sample of the more difficult problem distribution that can be expected in practical continuous-domain optimization. An ablation study is also performed on the LABCAT algorithm using this benchmark to determine the contribution of each significant element of the algorithm to overall performance. Using the results from these benchmarks, the performance of LABCAT relative to other algorithms and when applied to certain function groups with shared characteristics are discussed. All results in this dissertation were obtained using an 11th generation Intel i7–11700 CPU @ 2.5 GHz, with the exception of those extracted from the COCO archive.

## 10.1 Synthetic Test Functions Benchmark

As outlined in Section 1.1, standard BO has several shortcomings, the addressing of which forms the motivation for this research. This section will focus on two of these shortcomings, namely the poor convergence characteristics (such as the inability to converge arbitrarily close to a solution) and the tendency of standard BO to slow down computationally due to poor scaling with additional algorithm iterations. To investigate to what extent the shortcomings of standard BO are inherited or addressed, the proposed LABCAT algorithm and existing trust-region-based BO algorithms, previously discussed in the literature study of Chapter 2, are applied to a set of well-known, synthetic 2-D test functions with a range of objective function characteristics.

The set of selected synthetic test functions are chosen as the sphere [98], quartic [99],

Booth [100], Branin-Hoo [101], Rosenbrock [58] and Levy [102] functions,[1] with the respective definitions of these function provided in Table 10.1 and visualizations of each function given in Figure 10.1. These synthetic functions are each designed with certain properties and have been selected to cover a large range of problem characteristics: The sphere, quartic, Booth and Rosenbrock functions are unimodal, with the Booth and Rosenbrock functions being badly conditioned and the optimum of the Rosenbrock function being in a curved valley, making convergence to an arbitrary precision difficult. The Branin-Hoo and Levy functions are multimodal, with the Branin-Hoo function having several global minima and the Levy function having several local minima. The sphere, quartic and Booth functions are also separable, with the first two being separable along the coordinate axes while the latter is not. Note that, while these functions have closed-form definitions and, therefore, may have well-defined gradients, for the purposes of this analysis they are treated as black-box objective functions to simulate the problem of black-box optimization.

| Function | Definition | Domain | $f_{\min}$ |
|---|---|---|---|
| Sphere [98] | $f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2$ | $x_i \in [-5.12, 5.12]$ | $f(\mathbf{x}_{\min}) = 0$, $\mathbf{x}_{\min} = (0, \dots, 0)$ |
| Quartic [99] | $f(\mathbf{x}) = \sum_{i=1}^{d} i x_i^4$ | $x_i \in [-1.28, 1.28]$ | $f(\mathbf{x}_{\min}) = 0$, $\mathbf{x}_{\min} = (0, \dots, 0)$ |
| Booth [100] | $f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$ | $x_1, x_2 \in [-10, 10]$ | $f(\mathbf{x}_{\min}) = 0$, $\mathbf{x}_{\min} = (1, 3)$ |
| Rosenbrock [58] | $f(\mathbf{x}) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | $x_i \in [-5, 10]$ | $f(\mathbf{x}_{\min}) = 0$, $\mathbf{x}_{\min} = (1, \dots, 1)$ |
| Branin-Hoo [101] | $f(\mathbf{x}) = (x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6)^2 + 10(1 - \frac{1}{8\pi}) \cos(x_1) + 10$ | $x_1 \in [-5, 10]$, $x_2 \in [0, 15]$ | $f(\mathbf{x}_{\min}) = 0.397887$, $\mathbf{x}_{\min} = (-\pi, 12.275)$, $(\pi, 2.275)$, and $(9.42478, 2.475)$ |
| Levy [102] | $f(\mathbf{x}) = \sum_{i=1}^{d-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)] + \sin^2(\pi w_1) + (w_d - 1)^2 [1 + \sin^2(2\pi w_d)]$, where $w_i = 1 + \frac{x_i - 1}{4} \quad \forall i = \{1, \dots, d\}$ | $x_i \in [-10, 10]$ | $f(\mathbf{x}_{\min}) = 0$, $\mathbf{x}_{\min} = (1, \dots, 1)$ |

Table 10.1: Selected synthetic benchmark function definitions.

This experiment has been performed using a Python framework and the LABCAT algorithm has been implemented using Rust and tested using the included Python interface with the source code available in the `labcat` library.[2] For the other selected trust-region-based BO algorithms from Chapter 2, the SRSM implementation from the Python `bayesian-optimization`[3] library is used as well as the Python interfaces provided by the authors of the TuRBO, TRLBO, BADS and TREGO algorithms from the `turbo`,[4] `trlbo`,[5] `pybads`[6] and `trieste`[7] libraries, respectively. A purely random search (implemented using Python) as well as standard BO, also from the `bayesian-optimization` package, is also included as a performance baseline.

[1] The sphere, Rosenbrock and quartic functions are also known as the first, second and fourth De Jong functions [98].

[2] `https://github.com/esl-sun/LABCAT`

[3] `https://github.com/bayesian-optimization/BayesianOptimization`

[4] `https://github.com/uber-research/TuRBO`

[5] `https://github.com/agier9/TRLBO`

[6] `https://github.com/acerbilab/pybads`

[7] `https://github.com/secondmind-labs/trieste`

The parameters of the LABCAT algorithm are set within the recommended ranges ($\beta = 0.5, \rho = 7$ and $\sigma_\ell = 0.1$) with the recommended starting design of experiment (DoE) budget $(2d + 1)$. Each of the additional algorithms used in the comparison are also initialized with default parameters and a DoE budget of $2d+1$. The TuRBO algorithm is also used in the two configurations of the original paper: a single trust region ("TuRBO-1") and five parallel trust regions ("TuRBO-5"). Using these selected algorithms, 50 independent optimization runs are performed for each benchmark function per algorithm with a sampling budget of 150 objective function evaluations. The results of this comparison are given in Figure 10.2 with a statistical significance analysis of these results given in Table 10.2 and Appendix B.

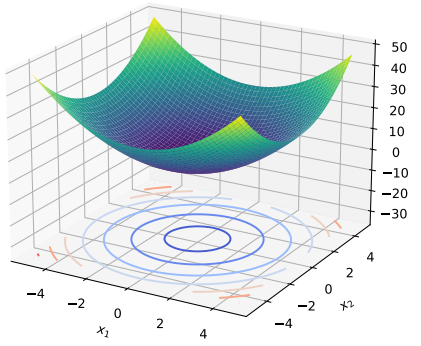
The noted convergence shortcomings of standard BO, namely that BO struggles to converge to an arbitrary precision, is clearly visible. This deficiency is also inherited by the SRSM, TuRBO and TRLBO algorithms, clearly making very slow progress after 150 objective function samples. The BADS algorithm exhibits better convergence characteristics for the Branin-Hoo and Levy before flattening out, possibly due to the deterministic mesh adaptive direct search (MADS) fallback step incorporated in this algorithm. It is clear that the LABCAT algorithm is not only capable of consistent convergence to a much higher level of precision for a wide range of objective function characteristics, but also does so at a faster overall rate than comparable trust-region-based BO and BO algorithms. It should be noted, however, that for several functions, the BADS and SRSM algorithms make faster initial progress for roughly the first 40 samples before being overtaken by the LABCAT algorithm.

To compare the real execution time needed and computational slowdown for a single run of each of the compared algorithms, the total wall-clock times for each of the algorithms are recorded and summarized in Table 10.2 and given for each function in Tables B.1 and B.2. To remove a potential source of bias, the time required to sample the objective function is not included in these measurements.

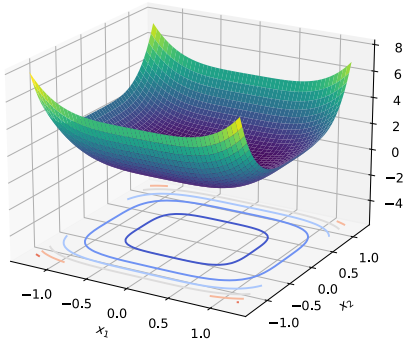
While the LABCAT algorithm exhibits the best average wall-clock time of the compared algorithms (Table 10.2), this should be interpreted as an indication of relative performance as these times do not account for differing implementations, use of multithreading and levels of code optimization for each algorithm. For example, use of a compiled language (Rust) versus an interpreted language (Python) could easily account for an order of magnitude difference in performance due to the additional code optimizations performed by the compiler.

For a more like-to-like comparison, the relative slowdown of each algorithm is measured as the difference between the average iteration time over the entire 150 sample iterations and the average iteration time over the last 30 iterations (representing the final 20% of the run). This difference is calculated for each optimization run, and the average values of the 300 runs are reported in Table 10.2. These results clearly indicate a significant slowdown in the standard BO algorithm, a shortcoming that is observed to a lesser extent in the other tested TR-BO algorithms. Notably, LABCAT is the only algorithm which does not exhibit statistically significant[8] slowdown compared to random sampling, indicating that this shortcoming is not inherited by the LABCAT algorithm.

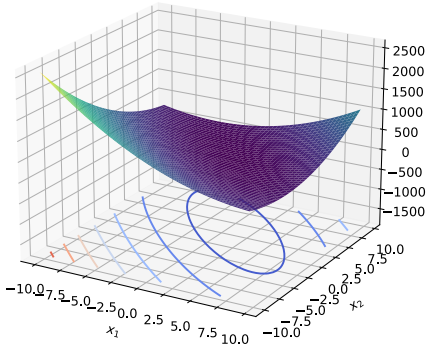
[8]Comparisons are considered to be statistically different from the results of the random sampling algorithm if the $p$-value obtained using Welch's t-test [103] is less than 0.05, adjusted using a Bonferroni correction [104] by the number of comparisons performed $\frac{0.05}{8} = 0.00625$ to reduce the false positive error rate.



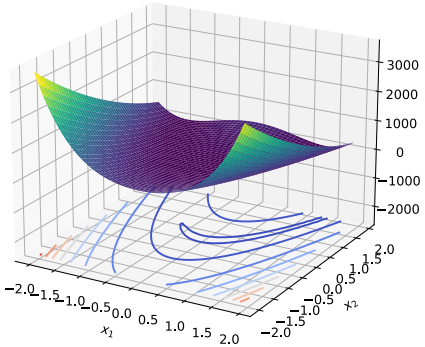

(a) Sphere



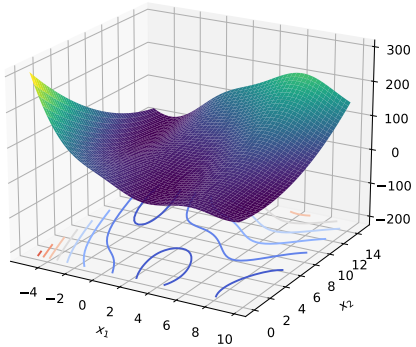

(b) Quartic



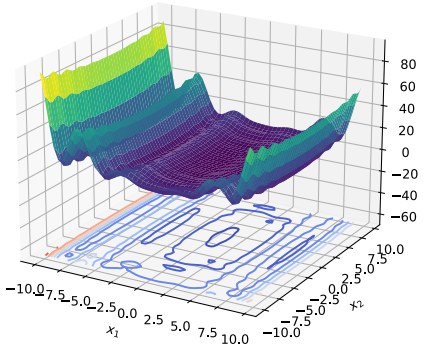

(c) Booth




(d) Rosenbrock

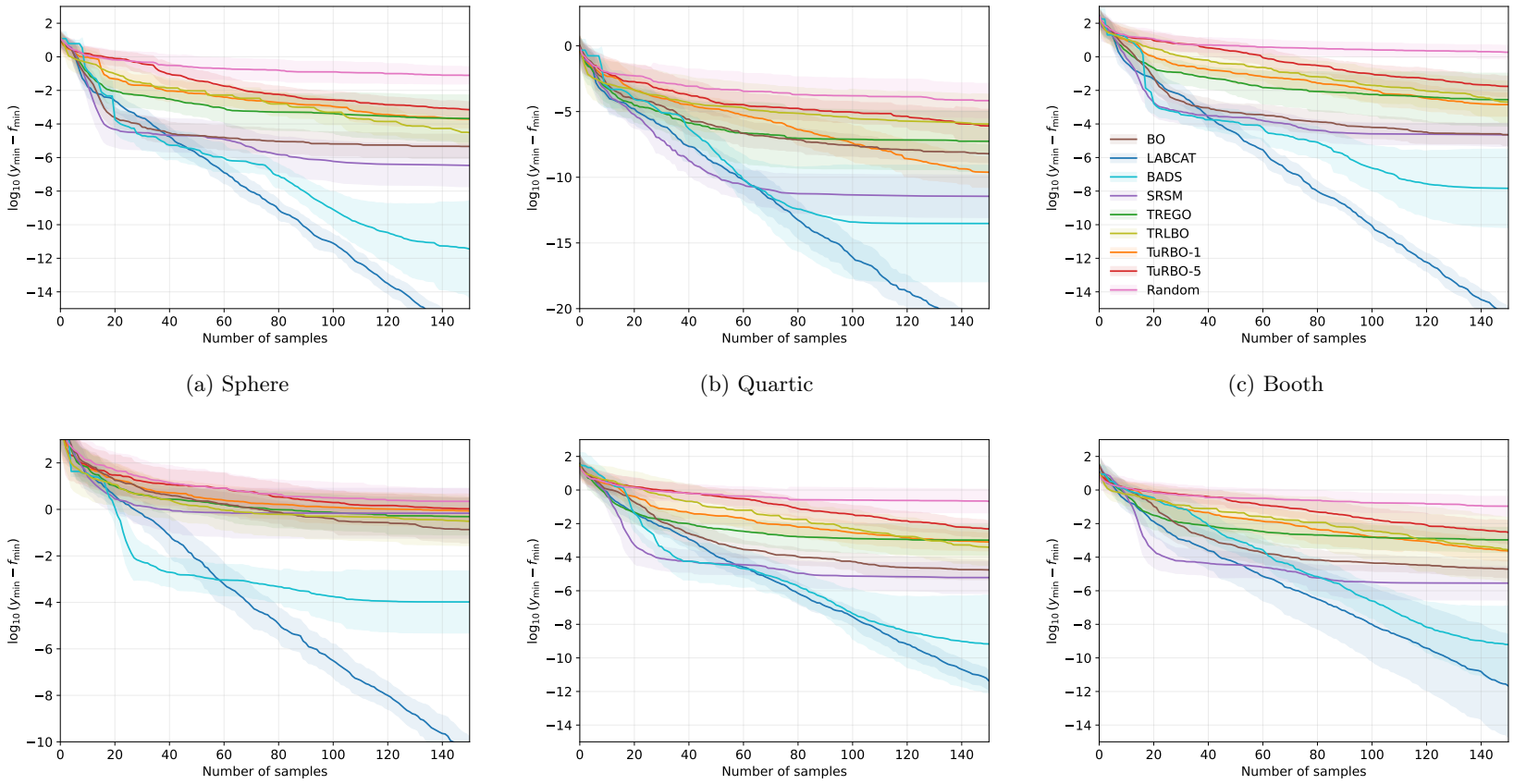



(e) Branin-Hoo




(f) Levy

Figure 10.1: Visualizations of the selected synthetic test functions from Table 10.1.




(a) Sphere (b) Quartic (c) Booth

(d) Rosenbrock (e) Branin-Hoo (f) Levy

Figure 10.2: Performance of selected algorithms applied to synthetic 2-D test functions. The mean and standard deviation, indicated by the shaded regions, of the logarithmic global regret, which is the log-difference between the best candidate solution $y_{\min}$ at each sampling iteration of the objective function and the global minimum $f_{\min}$, are reported. The definition and domain of each objective function is given in Table 10.1.

| Algorithm | Execution time | Iteration time ($t_{100\%}$) | Iteration time ($t_{20\%}$) | Change in average iteration times | $\uparrow/\approx/\downarrow$ |
|---|---|---|---|---|---|
| BO | 25.645 ± (1.239) | 1.721e−01 ± (8.257e−03) | 2.055e−01 ± (1.350e−02) | +19.544% ± (5.596%) | ↑ |
| LABCAT | **0.062 ± (0.022)** | **4.131e−04 ± (1.459e−04)** | **3.971e−04 ± (1.296e−04)** | **−0.225% ± (15.003%)** | ≈ |
| BADS | 5.696 ± (0.764) | 3.798e−02 ± (5.094e−03) | 4.323e−02 ± (1.171e−02) | +12.923% ± (23.749%) | ↑ |
| SRSM | 34.381 ± (1.958) | 2.292e−01 ± (1.305e−02) | 2.514e−01 ± (1.510e−02) | +10.545% ± (6.499%) | ↑ |
| TREGO | 110.821 ± (4.127) | 7.388e−01 ± (2.751e−02) | 7.830e−01 ± (4.826e−02) | +6.033% ± (6.254%) | ↑ |
| TRLBO | 1.380 ± (0.076) | 9.200e−03 ± (5.091e−04) | 9.615e−03 ± (1.098e−03) | +4.279% ± (8.051%) | ↑ |
| TuRBO-1 | 1.448 ± (0.141) | 9.654e−03 ± (9.387e−04) | 1.035e−02 ± (3.037e−03) | +5.738% ± (24.190%) | ↑ |
| TuRBO-5 | 3.936 ± (0.396) | 2.624e−02 ± (2.640e−03) | 2.956e−02 ± (5.857e−03) | +11.923% ± (17.258%) | ↑ |
| Random | 0.011 ± (0.004) | 7.378e−05 ± (2.579e−05) | 7.291e−05 ± (3.227e−05) | −0.959% ± (14.035%) | N/A |

Table 10.2: Average and standard deviations of the wall-clock times (in seconds) for a total of 300 independent optimization runs per selected algorithm over the 6 selected synthetic test functions from Table 10.2. The first column reports the total execution times. The second and third columns report the average time to complete a single algorithm iteration with the average measured over the entire optimization run ($t_{100\%}$) and the final 20% of the optimization run ($t_{20\%}$), respectively. The results in the final two columns indicate the observed difference in the average iteration times (calculated per optimization with the result averaged over all of the optimization runs) and whether this difference is statistically significantly[8] greater than, similar to, or less than (indicated using "↑", "≈" and "↓", respectively) the difference observed for the random sampling algorithm.

## 10.2 COCO Black-Box Optimization Benchmark

To further investigate the performance of the proposed LABCAT algorithm across a wider set of objective functions, the second experiment is performed using the comparing continuous optimizers (COCO) benchmarking software [40]. In this dissertation, the noiseless BBOB test suite [105] is used, which comprises 24 black-box objective functions to optimize, given in Table 10.3. The functions in this test suite are collected into 5 groups, each with the following shared characteristics: (i) separable, (ii) unimodal with moderate conditioning, (iii) unimodal with high conditioning, (iv) multimodal with adequate global structure, and (v) multimodal with weak global structure. In all, the results in this dissertation from the COCO software represent several weeks of combined CPU time on the previously mentioned hardware.

To evaluate the performance (as defined by the COCO software) of a single optimization algorithm on this suite, 15 *instances* of each function are generated, with each instance corresponding to a randomized modification of the function by a random translation of the optimum and a random rotation of the coordinate system. For each instance, an array of *problems* is also generated. Each of these problems are defined as a tuple comprising a function instance and a *target* precision to reach. To set these targets and give a good performance reference, COCO defines a composite algorithm known as best2009, an algorithm composed of the best performing optimization algorithm for each function from the BBOB-2009 workshop [106].[9] Using the COCO experimental setup for expensive objective functions (e.g., [48, 53, 54]), the targets for each instance are set to the values reached by best2009 after a certain number of objective function evaluations. Specifically, this number of function evaluations is set to 50 values $[0.5, \ldots, 100] \times d$, uniformly distributed in logarithmic space with the width of this distribution proportional to the objective function dimensionality $d$. For the analysis in this dissertation, all of the algorithms tested are provided with a total sampling budget of $200d$ objective function evaluations per function instance to reach the targets defined by best2009. If an algorithm terminates before exhausting the sampling budget, unless otherwise specified, an independent restart is performed with the remaining sampling budget.

With the recorded performance of an optimization algorithm applied to these problems, results known as runtime empirical cumulative distribution functions (runtime ECDFs) are compiled by the COCO software [40], which aggregates the proportion of problems solved by the algorithm for a given budget of objective function evaluations. Note that, similarly to the benchmarks in the previous section, a purely random search is also included in the analysis to serve as a rough lower bound for performance. An example of such a runtime ECDF generated by the COCO software is given in Figure 10.3, specifically for the performance of the LABCAT, best2009 and random algorithms applied to the 2-D Rosenbrock function ($f_8$ from Table 10.3). In this example figure, the traces of each tested algorithm show the proportion of previously defined, best2009-based optimization targets achieved (vertical axis) against the number of objective function evaluations (horizontal axis). Since targets cannot be unreached, the traces are monotonic. As such, an algorithm that reaches the targets faster using fewer evaluations will have a larger area under its curve. Therefore, both the area under the curve and the final proportion of targets achieved are valuable for assessing the relative performance of the algorithms, as the area under the curve reflects the rate at which the optimization targets are achieved, while the final value indicates how closely the algorithm

[9] Note that, due to being composed of the best performing algorithms for each function, outperforming best2009 is quite a difficult task.

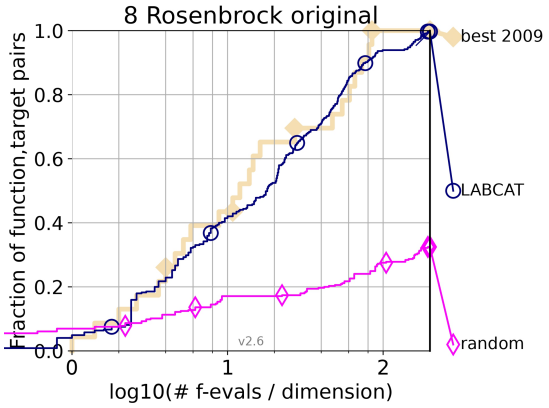



Figure 10.3: Example runtime empirical cumulative distribution function (ECDF) generated by the COCO software.

has approached the optimum. Also note that the horizontal axis is given as the logarithm of the number of objective function evaluations normalized by the problem dimensionality, in order to promote readability and ease of comparison between different problem dimensions.

## 10.3 LABCAT Ablation Study with the COCO Benchmark

During the development of the LABCAT algorithm, several design choices have been made, such as the choice of weights during the calculation of the weighted principal components (Section 7.2) or the choice of prior distribution during the determination of the most likely length-scales (Section 9.2.1). As such, it is necessary to verify these choices experimentally. Therefore, an ablation study is performed on the LABCAT algorithm to verify these design choices made during the development of the algorithm as well as to assess the contribution and significance of each component of the LABCAT algorithm on overall performance. In this analysis, a single component of the LABCAT algorithm is removed or modified in some way to either hamper or magnify its effect on the total performance of the algorithm. This ablation study is divided into two parts, where the first contains most of the ablated versions of the LABCAT algorithm and the second part investigates the choice of weight matrix $\mathbf{W}$.

For the first half of the analysis, the unablated, baseline algorithm is set as the full LABCAT algorithm with a set of parameters within the recommended ranges ($\beta = \frac{1}{d}, \rho = 7$ and $\sigma_\ell = 0.1$), denoted as "LABCAT", and compared against instances of LABCAT with (a) no principal component rotation ("LABCAT noPC"), (b) more passive discarding of observations by doubling the maximum recommended value for $\rho$ to 20 ("LABCAT p20"), (c) a uniform length-scale prior distribution instead of a multivariate Gaussian distribution ("LABCAT ULSP"), (d) an increased number of Newton or gradient steps during hyperparameter optimization from 1 to 10 ("LABCAT n10"), and (e) more rigorous acquisition function maximization using a more computationally intensive, multistart approach with the L-BFGS-B algorithm [8] with $10d$ starting points across the trust region ("LABCAT BFGS"). The results obtained by applying each of the ablated versions of LABCAT on the BBOB test suite is summarized for all test functions in Figure 10.4, with the full results for function groups with shared characteristics given in Appendix C.1.1.

| | Function name | Notes |
|---|---|---|
| | (i) Separable functions | |
| $f_1$ | Sphere | |
| $f_2$ | Ellipsoidal | Conditioning $\approx 10^6$ |
| $f_3$ | Rastrigin | $10^d$ local minima with regular structure |
| $f_4$ | Büche-Rastrigin | Asymmetric transform applied to $f_3$ |
| $f_5$ | Linear Slope | Linear function with solution on domain boundary |
| | (ii) Unimodal functions with low or moderate conditioning | |
| $f_6$ | Attractive Sector | Highly asymmetric |
| $f_7$ | Step Ellipsoidal | Similar to $f_2$, consisting of many plateaus |
| $f_8$ | Rosenbrock | Curved $n-1$ dimensional valley |
| $f_9$ | Rotated Rosenbrock | Rotated $f_8$ |
| | (iii) Unimodal functions with high conditioning | |
| $f_{10}$ | Ellipsoidal | Rotated $f_2$ |
| $f_{11}$ | Discus | Single search direction 1000 times more sensitive than all others |
| $f_{12}$ | Bent Cigar | Non-quadratic valley must be followed to optimum |
| $f_{13}$ | Sharp Ridge | Similar to $f_{12}$, non-differentiable valley floor |
| $f_{14}$ | Different Powers | Input variable sensitivities change approaching optimum |
| | (iv) Multimodal functions with adequate global structure | |
| $f_{15}$ | Rastrigin | Non-separable $f_3$ |
| $f_{16}$ | Weierstrass | Non-unique optimum, highly rugged and moderately repetitive landscape |
| $f_{17}$ | Schaffers F7 | Highly multimodal with varying frequency and amplitude of modulation |
| $f_{18}$ | Schaffers F7, Moderately ill-conditioned | Moderately ill-conditioned $f_{17}$ |
| $f_{19}$ | Composite Griewank-Rosenbrock | Resembles $f_8$ in a highly multimodal way |
| | (v) Multimodal functions with weak global structure | |
| $f_{20}$ | Schwefel | $2^d$ prominent local minima in corners of unpenalized, rectangular search area |
| $f_{21}$ | Gallagher's Gaussian 101-me Peaks | 101 optima with random heights and positions |
| $f_{22}$ | Gallagher's Gaussian 21-hi Peaks | 21 optima with random heights and positions, higher conditioning (1000) vs. $f_{21}$ (30) |
| $f_{23}$ | Katsuura | $10^d$ global optima, highly rugged and repetitive |
| $f_{24}$ | Lunacek bi-Rastrigin | Highly multimodal with 2 funnels, one leading to a local minimum that covers 70% of the search space |

Table 10.3: BBOB test suite objective functions. Notes regarding each objective function are given by the authors of the original test suite specification document by Hansen et al. [105].

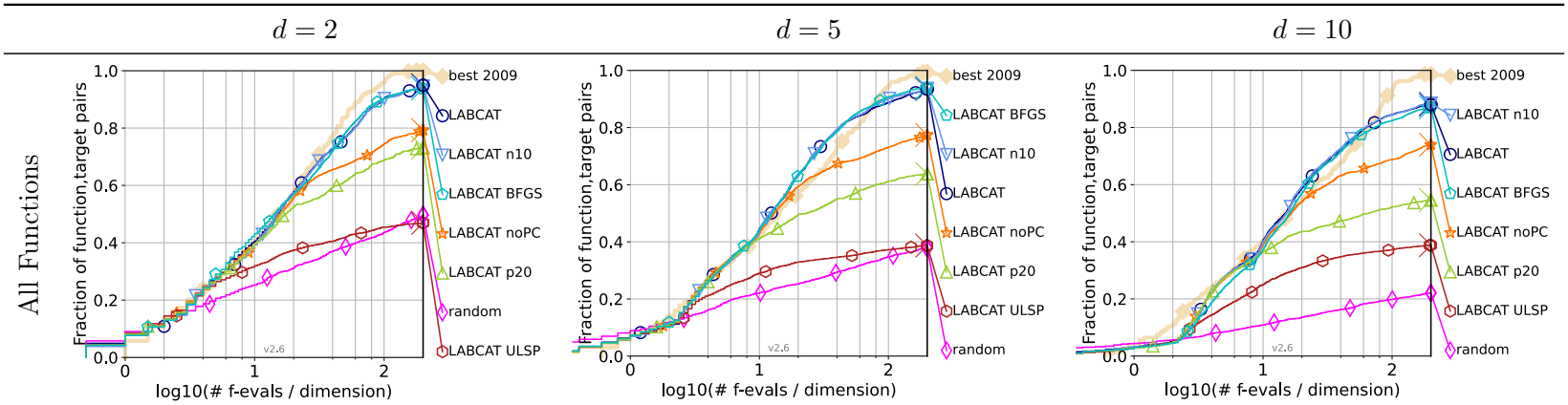



Figure 10.4: Empirical cumulative distribution functions (ECDFs) of runtimes table for the first half of the ablation study with the COCO dataset over all functions for dimensions $2, 5$ and $10$.

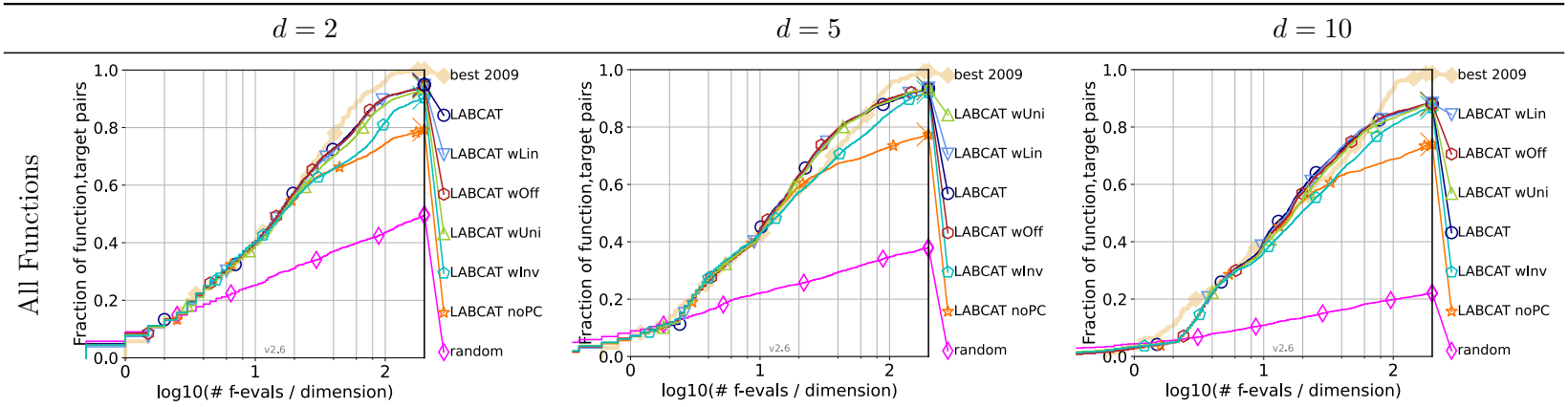



Figure 10.5: Empirical cumulative distribution functions (ECDFs) of runtimes table for the second half of the ablation study using different weight matrices with the BBOB test suite over all functions for dimensions $2, 5$ and $10$.

It is clear that the principal component rotation contributes significantly to the overall LABCAT algorithm performance, with the resulting performance decrease due to the removal thereof ("LABCAT noPC") especially visible in lower dimensions and for unimodal functions in groups (ii) and (iii). This aligns with the behaviour observed in the illustrative example of Figure 7.3, as several of the unimodal functions are characterized by valleys with changes in direction, similar to the Rosenbrock function, for which the principal components align the trust region with the direction of the valleys. A similar contribution can be seen by the observation discarding strategy ("LABCAT p20"), with a more pronounced difference in 5 and 10 dimensions. Severe performance degradation is observed when the Gaussian prior placed over the kernel length-scales in Equation 9.33 is replaced by a uniform distribution ("LABCAT ULSP"). While not indicated by the COCO-generated runtime ECDF graphs, a significant increase in the number of restarts was observed when using this uniform distribution, indicating instability in the ablated LABCAT algorithm. As opposed to the two previous modifications that may yield performance gains for certain function groups, replacing the length-scale prior is also a strict downgrade, with no performance gains in any function group. Additional Newton or gradient steps during the optimization of the length scales ("LABCAT n10") yield little to no significant performance increases, therefore a single Newton or gradient step seems to be sufficient, with all of the associated computational savings. Similarly, maximizing the expected improvement acquisition function using the more intensive multistart method ("LABCAT BFGS") also does not provide noticeable performance improvements and the method of using random sampling to maximize the expected improvement is deemed to be sufficient.

Inspecting Section C.1.1, it is interesting to note that removal of the principal component rotation and slower observation removal yields modest improvements when applied to the multimodal function groups (iv) and (v). This may be due to the slower convergence of these modified versions of the LABCAT algorithm, leading to more exploration of the objective function space and finding slightly better, hard to reach solutions for these functions. In practice, if meta-information regarding the objective function is available that indicates a multimodal structure and the additional computational cost can be spared, the observation cache multiplier $\rho$ could be increased for better performance.

In summary, from the results of the first half of the ablation study, each of the constituent components of LABCAT contribute significantly to overall performance and the assumptions made in Section 9.2 and 9.3 are shown to be well-founded.

Having analyzed the relative contributions of several components of the LABCAT algorithm, the second half of the ablation study is performed to determine the effect of the choice of weights $\mathbf{W}$ applied to each observed input for the determination of weighted principal components in Section 7.2 and 9.1. As these weights determine the relative contribution of each observation to the "important directions" identified in the observed data (in effect, the principal components) with no proven, universally optimal choice for these weights, it is prudent to investigate if the principal components determined using one set of weights leads to improved performance compared using another. Therefore, for this half of the ablation study, instances of LABCAT are tested using (a) uniform weights, $W_{ii} = 1$ ("LABCAT wUni"), (b) quadratic weights with offset to ensure weight of $y_{\max}$ is nonzero, $W_{ii} = 0.9(1 - y_i) + 0.1$ ("LABCAT wOff"), (c) linear weights, $W_{ii} = \sqrt{1 - y_i}$ ("LABCAT wLin"), (d) inverse weights, $W_{ii} = y_i$ ("LABCAT wInv") as well as including the default "LABCAT" and "LABCAT noPC" as defined in the previous experiment. The results obtained by applying each of the versions of LABCAT with different weight values on the BBOB test suite is summarized in Figure 10.5

with the full results provided in Section C.1.2.

From Figure 10.5, it is clear that the performance of the LABCAT algorithm does improve when incorporating the output-based sample-wise weights when compared to uniform weights, although not as much as compared to no principal-component-based rotation whatsoever. The algorithm does not seem very sensitive to the exact choice of method to calculate $\mathbf{W}$, suggesting that a sufficient requirement may be to have higher relative weight to lower observations. Interestingly, using a rotation with inverse weights (assigning higher weights to observations with larger output values) also yields improved performance compared to using no rotation at all. This may be due to this rotation aligning the trust region with directions *orthogonal* to the local axes of separability, which, due to the rectangular shape of the trust region, may somewhat align secondary axes of the trust region with the separability axes.

## 10.4 Comparison with State-Of-The-Art Derivative-Free Optimization Algorithms using the COCO Benchmark

To compare the proposed LABCAT algorithm to algorithms from the wider field of derivative-free optimization when applied to the problem of black-box optimization, the set of algorithms included in the comparison from Section 10.1 is expanded with the state-of-the-art DTS-CMA-ES, AutoSAEA, MCS, NEWUOA and SMAC algorithms, similar to the comparisons performed by Acerbi and Ma [54] and Diouane et al. [53]. The DTS-CMA-ES [48] and AutoSAEA [49] algorithms use a surrogate-assisted evolutionary strategy based on a combination of the evolutionary algorithm and GP surrogates, known to be well-suited to multimodal problems. SMAC [22] is a variation of standard BO using an isotropic GP kernel and a locally biased stochastic search to optimize the expected improvement acquisition function. MCS [107] balances a global search based on the DIRECT [108] algorithm and a local search using a local quadratic interpolation. NEWUOA [109] also uses a quadratic interpolation, but combines it with a classical trust-region-based approach.

Results for DTS-CMA-ES, MCS, NEWUOA and SMAC were obtained from the COCO database (see the respective publications [110–113]). The results obtained by applying each of the functions on the BBOB test suite are shown in Figure 10.6, presenting results for 2, 3 and 5 dimensions across all functions and from the two most difficult function groups: (iii) unimodal functions with high conditioning and (v) multimodal functions with weak global structure, respectively. Complete results that include the other function groups can be found in Section C.2. Note that wall-clock times are not reported for this analysis, as this information is not recorded by the COCO software.

The results obtained from applying the selected algorithms to the BBOB test suite reveal several important findings. Firstly, the LABCAT algorithm emerges as the top performer when considering all of the tested functions, having, from inspection, the largest area under the curve for any single algorithm for all dimensions (excluding the composite best2009 algorithm). Additionally, the algorithm excels particularly well when applied to unimodal functions with high conditioning from group (iii), even surpassing best2009 for this function group that is not traditionally considered to be well-suited for BO and trust-region-based BO methods (a fact also reflected in the results of these algorithms applied to this function group). Furthermore, the LABCAT algorithm proves to be highly proficient for 2-D and 5-D, for which it achieves the best performance of all of the BO-based algorithms, with a smaller performance gap for 10-D

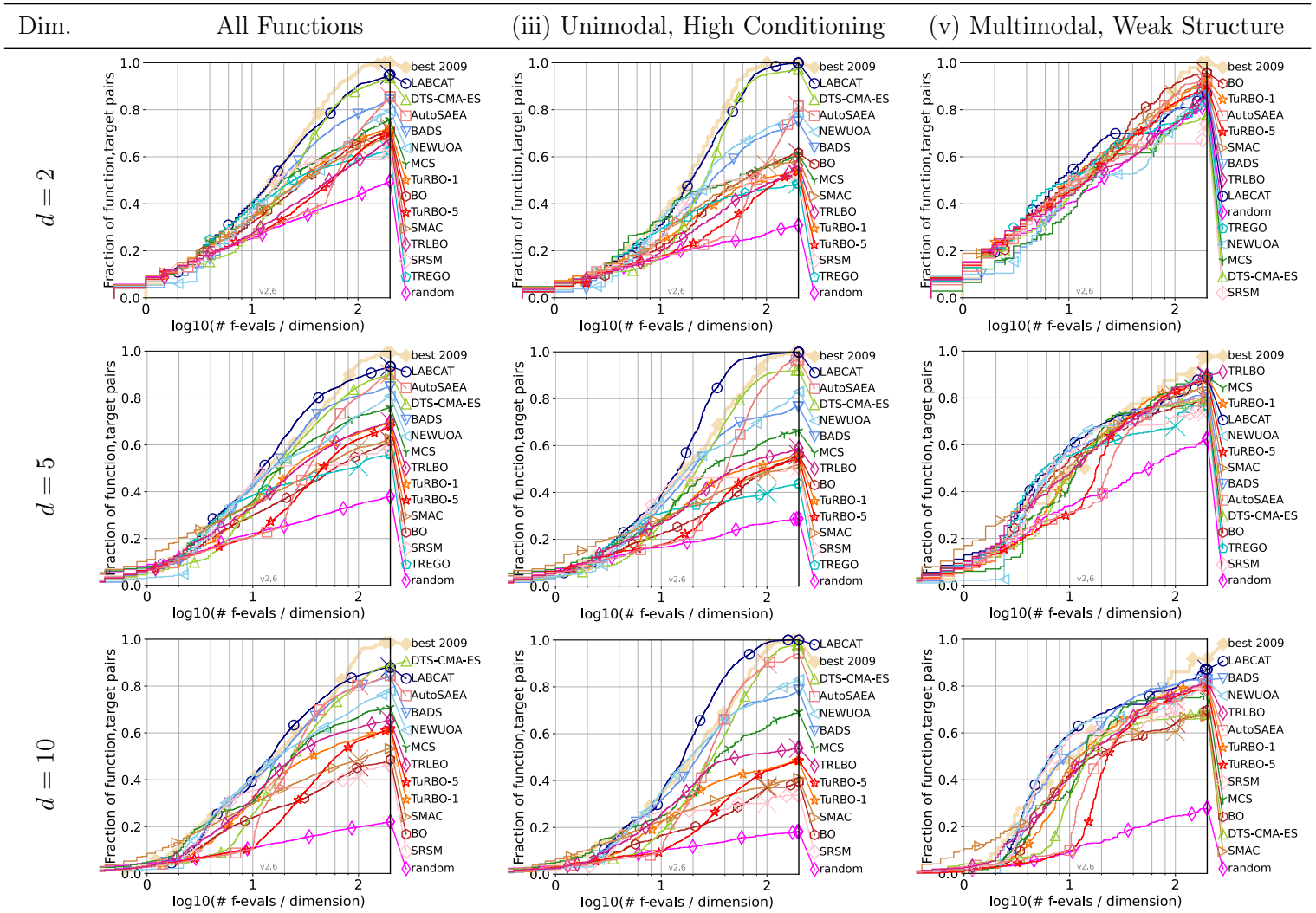



Figure 10.6: Selected empirical cumulative distribution functions (ECDFs) of runtimes table from the COCO benchmark for comparison of the LABCAT algorithm with various state-of-the-art optimization algorithms for dimensions $2, 5$ and $10$.

between the LABCAT and BADS algorithms. Upon inspection of the results in Section C.2, the performance of the LABCAT algorithm is also observed to be slightly lower than those of other trust-region-based BO algorithms (BADS, SRSM, TuRBO, TREGO) when applied to multimodal functions with weak global structure from function group (iv), probably due to LABCAT heavily favouring local exploration of the objective function and, by extension, being unable to model the underlying global structure of these functions as well as the other algorithms. This reduction in performance, however, can be mostly remedied by increasing the observation cache size factor $\rho$, as noted in the ablation study of the previous section.

The closest competitor from the compared trust-region-based BO algorithms, when considering all functions, is the BADS algorithm. The BADS algorithm is, however, not very resistant to highly conditioned functions from group (iii), being consistently outperformed by the LABCAT algorithm. It is unclear how much of the performance of the BADS algorithm can be ascribed to its deterministic MADS fallback step, although some information may be gleaned by comparing this performance to another trust-region-based BO method without this fallback step, such as TuRBO. In this comparison, similar performance is observed for multimodal functions, while BADS performs better for separable and unimodal functions. This may imply that the MADS fallback step incorporated into the BADS algorithm allows for increased local exploitation when compared to other, "purer" trust-region-based BO algorithms.

Considering the full set of tested algorithms, the only algorithm that approaches the performance of the LABCAT algorithm, and slightly outperforms it for 10-D, for all functions is the DTS-CMA-ES algorithm. This algorithm ends essentially tied with LABCAT for 2-D and slightly ahead for 10-D, although the LABCAT algorithm still has a larger area under the curve in both of these cases, indicating that the LABCAT algorithm achieved the required optimization targets earlier. Although the DTS-CMA-ES algorithm exhibits a somewhat slower start, it makes significant progress in subsequent iterations. From Section C.2, DTS-CMA-ES also performs noticeably better than the rest of the algorithms for multimodal functions with adequate global structure from group (iv), with this performance gap growing with dimension. This implies that DTS-CMA-ES is the algorithm that can leverage the underlying structure the most to ignore local minima.

Other observations of note include that SMAC seems to have an advantage for a very limited number of objective function evaluations before being overtaken, presumably due to SMAC being able to start optimizing before other algorithms have finished sampling their respective initial DoEs. The MCS algorithm performs notably well for separable functions, possibly due to the deterministic nature of the search that is aligned with the separability axes. The AutoSAEA algorithm makes significant progress after a slower start, similar to DTS-CMA-ES, due to the need for sampling the larger initial population required for the evolutionary framework.

In summary, the extent to which the proposed LABCAT algorithm addresses the noted shortcomings of standard BO has been presented in this chapter. Firstly, using a set of synthetic test functions, the LABCAT algorithm is shown to be able to converge to an arbitrary precision without experiencing computational slowdown, in contrast to standard BO and, to a lesser extent, trust-region-based BO methods. Secondly, using the COCO software and BBOB test suite, the LABCAT algorithm is shown to be a leading contender in the field of expensive black-box function optimization, performing better than all of the considered BO and trust-region-based BO methods when considering all of the BBOB functions, with the exception of

being tied for 10-D with the BADS algorithm. The LABCAT algorithm also performs notably well for unimodal and highly conditioned functions, functions not normally associated with standard BO.

# Chapter 11

# Conclusion

Standard Bayesian optimization (BO) is a popular and well-studied technique for the optimization of black-box objective functions, especially renowned for a high degree of sample efficiency, which allows it to find good solutions with relatively few objective function evaluations. Unfortunately, this method has several notable shortcomings which hinder its use across a broad range of optimization problems and objective functions. Specifically, it can experience significant computational slowdown as the number of algorithm iterations increases, which poses challenges for real-time applications. Additionally, BO may struggle with non-stationary and ill-conditioned objective functions due to the reliance on a kernel-based Gaussian process (GP) surrogate model that often requires manual kernel engineering to model these objective functions adequately. Finally, BO can exhibit poor convergence characteristics, due to a lack of theoretical guarantees and practical, numerical limitations.

This chapter contains an evaluation of the LABCAT algorithm in the context of the research aims and objectives outlined in Chapter 1. Additionally, the key contributions of this dissertation are highlighted and discussed, which seek to address the noted shortcomings of standard BO. Finally, possible future avenues for research are outlined.

## 11.1 Evaluation of the LABCAT Algorithm

To address the noted shortcomings of standard BO, a set of research objectives were defined in Section 1.2: (1) to investigate other existing modified BO methods to identify a broad strategy of the proposed algorithm, (2) to derive a novel algorithm that addresses the noted shortcomings of standard BO, and (3) to test this proposed algorithm using extensive and representative numerical benchmarks.

Firstly, several approaches that have been proposed to address these shortcomings of standard BO by incorporating some measure of local focus were investigated, with the broad outlines of these approaches discussed in Chapter 2. Of these approaches, trust-region-based Bayesian optimization (BO) algorithms were identified as partially addressing these shortcomings by constraining the selection of the next point to be sampled from the objective function and incorporated into the Gaussian process (GP) surrogate model using an iteratively updated trust region. This constraint ensures that the algorithm focuses its sampling efforts within a local area where the surrogate model may be more accurate, improving performance. By iteratively updating the trust region based on successive observations, trust-region-based

BO algorithms offer a more targeted optimization strategy, potentially mitigating issues related to computational slowdown and convergence performance.

Building on the trust-region-based BO approach, the LABCAT algorithm was derived and presented in Chapter 6–9. This proposed algorithm extends trust-region-based BO with the addition of an adaptive observation and trust-region rescaling strategy, based on the length-scales of the local GP surrogate, and rotation strategy, based on the weighted principal components of the observations. Combined with a greedy observation discarding strategy and approximative hyperparameter estimation, these components allow the LABCAT algorithm to address the noted shortcomings of BO.

To verify the extent to which the shortcomings of BO are addressed, a set of numerical benchmarks were performed in Chapter 10. Using a set of diverse synthetic test functions, a comparison of the proposed LABCAT algorithm with standard BO and a variety of state-of-the-art trust-region-based BO algorithms shows that the LABCAT algorithm is capable of convergence to a much higher level of precision without encountering numerical issues or instability. An analysis of the execution times for these synthetic test functions also show that the LABCAT algorithm does not inherit the computational slowdown of standard BO.

A second comparison with a range of state-of-the-art black-box optimization methods from the wider field of black-box optimization, performed using the COCO benchmarking software and BBOB test suite, shows that the LABCAT algorithm is a strong contender in the domain of expensive black-box function optimization, significantly outperforming standard BO for nearly all tested scenarios and demonstrating exceptional performance compared to state-of-the-art black-box optimization methods, particularly in the domain of unimodal and highly conditioned objective functions not typically associated with BO. A small performance gap remains for higher-dimensional, multimodal functions with global structure, possibly due to the increased emphasis of the LABCAT algorithm on local exploitation and a lack of a global surrogate model. While this can be remedied to an extent by setting the observation cache size factor $\rho$ to a larger value, it is important to note that the structure of the LABCAT algorithm does not preclude the use of existing techniques such as multiple parallel trust regions [51] or maintaining both a local and global surrogate model [53]. The potential of these techniques when applied to the LABCAT algorithm, however, remains to be explored.

## 11.2 Contributions

Specifically, the following original contributions made in this dissertation are enumerated as:

1. The LABCAT algorithm: a combination of two novel extensions of trust-region-based BO with a greedy observation discarding strategy and an approximative hyperparameter estimation scheme in a trust-region-based BO framework. Through extensive numerical experiments using a set of synthetic test functions and the well-known COCO benchmarking software, the LABCAT algorithm is shown to outperform several state-of-the-art BO and other black-box optimization algorithms.

2. An extension of trust-region-based BO which utilizes a novel rotation of the trust region, based on the weighted principal components of the observed input values: This rotation allows dynamic alignment of the trust region with local axes of separability in arbitrary directions, not just limited to the directions defined by the coordinate axes as previously

used by Eriksson et al. [51] and Li et al. [55]. This rotation is also in contrast to those found in classical trust-region-based, which are based on the Hessian of the objective function, information not available for black-box problems. As illustrated in Section 7.3 and observed in the results of Section 10.2, aligning the trust region with these local axes of separability allows for the expansion and contraction of the trust region along these directions, improving performance and allowing adaptation to large, ill-conditioned valleys that often hinder other BO and trust-region-based BO methods.

3. A second extension of trust-region-based BO using a novel length-scale-based rescaling of the observed inputs: This rescaling induces a dynamic trust region update strategy based on the local landscape of the objective function, captured by the value of the inferred length-scales, that can readily adapt to non-stationary and ill-conditioned functions which have proven to be difficult for standard BO methods. This dynamic trust region update strategy stands in contrast to existing trust-region-based BO methods that use fixed heuristics to update the trust region. Combined with the discarding of observations outside of the trust region, this rescaling also allows for convergence much closer to a solution before encountering the numerical issues faced by standard BO.

In summary, these contributions address the noted shortcomings of BO, allowing for the application of this algorithm to a broader class of objective functions and optimization problems, while maintaining the sample efficiency of standard BO.

## 11.3 Future Work

While the focus of this research has been the optimization of objective functions that are (i) black-box functions with (ii) continuous input parameters and (iii) that are observable exactly (i.e., with no noise), an important avenue for future work may include extending the LABCAT algorithm for use with a more general class of objective functions.

For example, to be applied to noisy objective functions with unknown noise profiles (often found in real-world sensor data), some way to estimate the noise variance would be required, possibly based on a normality test such as the Shapiro-Wilk test [114]. Categorical- and integer-valued input values (commonly encountered in the hyperparameters of machine learning models or when comparing a set of models) could be incorporated using the kernel modification technique proposed by Garrido-Merchán and Hernández-Lobato [115]. The GP used in the LABCAT algorithm could also be augmented with gradient observations [30, Ch. 9] to improve the quality of the surrogate model and speed of convergence, possibly allowing for competitive performance when applied to non-black-box optimization problems.

The potential of the LABCAT framework for use in high-dimensional problems also remains to be explored, possibly through the substitution of the GP surrogate model for computationally cheaper (but less accurate) surrogate models such as random forests [22] or tree-structured Parzen estimators [68]. The use of the weighted principal components to align the trust region with important search directions may also be of use to prioritize sampling efforts in these directions or simplify the trust region update strategy to these directions. As stated previously, since the LABCAT algorithm is a local optimizer (and can be aggressively so by setting the trust region size factor $\beta$ to a small value), the use of multistarts or parallel surrogate models (such as those used by Eriksson et al. [51]) for optimizing high-dimensional, multimodal problems also remain to be investigated.

# Bibliography


[1] P. de Fermat, "Methodus ad disquirendam maximam et minimam," in *Varia opera mathematica D. Petri de Fermat, senatoris Tolosani*, pp. 63–73, Johannes Pech, 1679.

[2] J. Wallis, "A treatise of algebra, both historical and practical," *Philosophical Transactions of the Royal Society of London*, vol. 15, no. 173, pp. 1095–1106, 1685.

[3] D. Fowler and E. Robson, "Square root approximations in old Babylonian mathematics: YBC 7289 in context," *Historia Mathematica*, vol. 25, no. 4, pp. 366–378, 1998.

[4] R. Priem, H. Gagnon, I. Chittick, S. Dufresne, Y. Diouane, and N. Bartoli, "An efficient application of Bayesian optimization to an industrial MDO framework for aircraft design," in *AIAA AVIATION 2020 Forum*, Jun 2020.

[5] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

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

[7] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans, "Automatic gait optimization with Gaussian process regression," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI'07, (San Francisco, CA, USA), pp. 944–949, Morgan Kaufmann Publishers Inc., 2007.

[8] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, pp. 1190–1208, 1995.

[9] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[10] G. Venter, *Review of Optimization Techniques*. John Wiley & Sons, Ltd, 2010.

[11] A. Rodomanov and Y. Nesterov, "Rates of superlinear convergence for classical quasi-Newton methods," *Mathematical Programming*, vol. 194, pp. 159–190, Jul 2022.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[13] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[14] M. J. D. Powell, "A hybrid method for nonlinear equations," *Numerical Methods for Nonlinear Algebraic Equations*, pp. 87–161, 1970.

[15] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*. Society for Industrial and Applied Mathematics, 2000.

[16] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, 2009.

[17] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, pp. 308–313, Jan 1965.

[18] R. Hooke and T. A. Jeeves, ""Direct Search" solution of numerical and statistical problems," *Journal of the ACM*, vol. 8, pp. 212–229, Apr 1961.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4, 1995.

[20] M. Pincus, "A Monte Carlo method for the approximate solution of certain types of constrained optimization problems," *Operations Research*, vol. 18, no. 6, pp. 1225–1228, 1970.

[21] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[22] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization* (C. A. C. Coello, ed.), (Berlin, Heidelberg), pp. 507–523, Springer Berlin Heidelberg, 2011.

[23] J. Bossek, C. Doerr, and P. Kerschke, "Initial design strategies and their effects on sequential model-based optimization: an exploratory case study based on BBOB," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, (New York, NY, USA), pp. 778–786, Association for Computing Machinery, 2020.

[24] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, pp. 148–175, 2016.

[25] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[26] D. Huang, T. Allen, W. Notz, and N. Zeng, "Global optimization of stochastic black-box systems via sequential Kriging meta-models," *Journal of Global Optimization*, vol. 34, pp. 441–466, Mar 2006.

[27] R. Garnett, *Bayesian Optimization*. Cambridge University Press, 2023.

[28] R. Couperthwaite, A. Molkeri, D. Khatamsaz, A. Srivastava, D. Allaire, and R. Arròyave, "Materials design through batch Bayesian optimization with multisource information fusion," *JOM*, vol. 72, pp. 4431–4443, Dec 2020.

[29] K. Wang and A. W. Dowling, "Bayesian optimization for chemical products and functional materials," *Current Opinion in Chemical Engineering*, vol. 36, p. 100728, 2022.

[30] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series, MIT Press, 2005.

[31] G. Lan, J. M. Tomczak, D. M. Roijers, and A. Eiben, "Time efficiency in optimization with a Bayesian-evolutionary algorithm," *Swarm and Evolutionary Computation*, vol. 69, p. 100970, 2022.

[32] J. Quiñonero-Candela, C. E. Rasmussen, and C. K. I. Williams, "Approximation methods for Gaussian process regression," in *Large-Scale Kernel Machines*, pp. 203–224, MIT Press, 08 2007.

[33] R. Martinez-Cantin, "Funneled Bayesian optimization for design, tuning and control of autonomous systems," *IEEE Transactions on Cybernetics*, vol. 49, pp. 1489–1500, Apr 2019.

[34] E. Vazquez and J. Bect, "Convergence properties of the expected improvement algorithm with fixed mean and covariance functions," *Journal of Statistical Planning and Inference*, vol. 140, no. 11, pp. 3088–3095, 2010.

[35] A. D. Bull, "Convergence rates of efficient global optimization algorithms.," *Journal of Machine Learning Research*, vol. 12, no. 10, 2011.

[36] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), pp. 1015–1022, Omnipress, 2010.

[37] R. B. Gramacy and H. K. H. Lee, "Cases for the nugget in modeling computer experiments," *Statistics and Computing*, vol. 22, pp. 713–722, 2010.

[38] T. Santner, B. Williams, and W. Notz, *The Design and Analysis of Computer Experiments*. Springer Series in Statistics, New York, NY: Springer, Jan 2018.

[39] M. McLeod, S. Roberts, and M. A. Osborne, "Optimization, fast and slow: optimally switching between local and Bayesian optimization," in *International Conference on Machine Learning*, pp. 3443–3452, PMLR, 2018.

[40] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, vol. 36, pp. 114–144, 2021.

[41] H. Mohammadi, R. Le Riche, and E. Touboul, "Making EGO and CMA-ES complementary for global optimization," in *Learning and Intelligent Optimization* (C. Dhaenens, L. Jourdan, and M.-E. Marmion, eds.), (Cham), pp. 287–292, Springer International Publishing, 2015.

[42] K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Perez, "Bayesian optimization with exponential convergence," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, vol. 2 of *NIPS'15*, (Cambridge, MA, USA), pp. 2809–2817, MIT Press, 2015.

[43] Z. Wang, B. Shakibi, L. Jin, and N. de Freitas, "Bayesian multi-scale optimistic optimization," in *Artificial Intelligence and Statistics*, pp. 1005–1014, PMLR, 2014.

[44] K. P. Wabersich and M. Toussaint, "Advancing Bayesian optimization: The mixed-global-local (MGL) kernel and length-scale cool down," in *Workshop on Bayesian Optimization*, Neural Information Processing Systems, 2016.

[45] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[46] K. S. Anderson and Y. Hsu, "Genetic crossover strategy using an approximation concept," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, Washington, DC, USA July 6-9, 1999*, pp. 527–533, IEEE, 1999.

[47] K. Abboud and M. Schoenauer, "Surrogate deterministic mutation: Preliminary results," in *Artificial Evolution* (P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, eds.), (Berlin, Heidelberg), pp. 104–116, Springer Berlin Heidelberg, 2002.

[48] L. Bajer, Z. Pitra, J. Repický, and M. Holeňa, "Gaussian process surrogate models for the CMA evolution strategy," *Evolutionary Computation*, vol. 27, pp. 665–697, Dec 2019.

[49] L. Xie, G. Li, Z. Wang, L. Cui, and M. Gong, "Surrogate-assisted evolutionary algorithm with model and infill criterion auto-configuration," *IEEE Transactions on Evolutionary Computation*, vol. 28, pp. 1114–1126, Jul 2023.

[50] N. Stander and K. Craig, "On the robustness of a simple domain reduction scheme for simulation-based optimization," *International Journal for Computer-Aided Engineering and Software (Engineering Computations)*, vol. 19, pp. 431–450, Jun 2002.

[51] D. Eriksson, M. Pearce, J. R. Gardner, R. Turner, and M. Poloczek, "Scalable global optimization via local Bayesian optimization," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, (Red Hook, NY, USA), pp. 5496–5507, Curran Associates Inc., 2019.

[52] R. G. Regis, "Trust regions in Kriging-based optimization with expected improvement," *Engineering Optimization*, vol. 48, no. 6, pp. 1037–1059, 2016.

[53] Y. Diouane, V. Picheny, R. L. Riche, and A. S. D. Perrotolo, "TREGO: a trust-region framework for efficient global optimization," *Journal of Global Optimization*, vol. 86, pp. 1–23, 2021.

[54] L. Acerbi and W. J. Ma, "Practical Bayesian optimization for model fitting with Bayesian adaptive direct search," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1834–1844, 2017.

[55] Q. Li, A. Fu, W. Wei, and Y. Zhang, "A trust region based local Bayesian optimization without exhausted optimization of acquisition function," *Evolving Systems*, vol. 14, pp. 839–858, Oct 2023.

[56] D. J. C. MacKay, "Bayesian methods for backpropagation networks," in *Models of Neural Networks III: Association, Generalization, and Representation*, pp. 211–254, New York, NY: Springer New York, 1996.

[57] R. M. Neal, *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics, New York, NY: Springer, 1996.

[58] H. H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *The Computer Journal*, vol. 3, pp. 175–184, Jan 1960.

[59] C. Audet and J. E. Dennis, "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188–217, 2006.

[60] D. J. C. MacKay, "Introduction to Gaussian processes," *NATO ASI series. Series F: computer and system sciences*, pp. 133–165, 1998.

[61] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 2nd ed., 2012.

[62] Benoit, "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés a un système d'équations linéaires en nombre inférieur a celui des inconnues (procédé du commandant Cholesky)," *Bulletin Géodésique*, vol. 2, pp. 67–77, Apr 1924.

[63] C. N. Haddad, "Cholesky factorization," in *Encyclopedia of Optimization* (C. A. Floudas and P. M. Pardalos, eds.), pp. 374–377, Boston, MA: Springer US, 2009.

[64] J. Mercer, "Functions of positive and negative type, and their connection with the theory of integral equations," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 209, pp. 415–446, 1909.

[65] F. Vivarelli and C. Williams, "Discovering hidden features with Gaussian processes regression," in *Advances in Neural Information Processing Systems* (M. Kearns, S. Solla, and D. Cohn, eds.), vol. 11, MIT Press, 1998.

[66] C. Williams and C. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems* (D. Touretzky, M. Mozer, and M. Hasselmo, eds.), vol. 8, MIT Press, 1995.

[67] R. M. Neal, "Monte Carlo implementation of Gaussian process models for Bayesian regression and classification," Tech. Rep. GR-G-9702, University of Toronto, 1997.

[68] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, (Red Hook, NY, USA), pp. 2546–2554, Curran Associates Inc., 2011.

[69] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization," *Journal of Machine Learning Research*, vol. 13, pp. 1809–1837, Jun 2012.

[70] P. I. Frazier, W. B. Powell, and S. Dayanik, "A knowledge-gradient policy for sequential information collection," *SIAM Journal on Control and Optimization*, vol. 47, no. 5, pp. 2410–2439, 2008.

[71] D. Zhan and H. Xing, "Expected improvement for expensive optimization: a review," *Journal of Global Optimization*, vol. 78, pp. 507–544, Nov 2020.

[72] F. Zhang, *The Schur complement and its applications*. Numerical Methods and Algorithms, Springer, 2005.

[73] M. D. McKay, R. Beckman, and W. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, pp. 239–245, May 1979.

[74] R. Fletcher, "Restricted step methods," in *Practical Methods of Optimization*, ch. 5, pp. 95–109, John Wiley & Sons, Ltd, 2nd ed., 1981.

[75] D. C. Sorensen, "Newton's method with a model trust region modification," *SIAM Journal on Numerical Analysis*, vol. 19, no. 2, pp. 409–426, 1982.

[76] H. Nielsen, "Damping parameter in Marquardt's method," Tech. Rep. IMM-REP-1999-05, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 1999.

[77] D. Calandriello, L. Carratino, A. Lazaric, M. Valko, and L. Rosasco, "Scaling Gaussian process optimization by evaluating a few unique candidates multiple times," in *International Conference on Machine Learning*, pp. 2523–2541, PMLR, 2022.

[78] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[79] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, pp. 498–520, 1933.

[80] I. T. Jolliffe, *Principal Component Analysis*. Springer Series in Statistics, New York: Springer-Verlag, 2 ed., 2002.

[81] G. W. Stewart, "On the early history of the singular value decomposition," *SIAM Review*, vol. 35, no. 4, pp. 551–566, 1993.

[82] M. Greenacre, *Theory and Applications of Correspondence Analysis*. Academic Press, 1984.

[83] S. Bailey, "Principal component analysis with noisy and/or missing data," *Publications of the Astronomical Society of the Pacific*, vol. 124, no. 919, p. 1015, 2012.

[84] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "A general framework for increasing the robustness of PCA-based correlation clustering algorithms," in *Scientific and Statistical Database Management* (B. Ludäscher and N. Mamoulis, eds.), (Berlin, Heidelberg), pp. 418–435, Springer Berlin Heidelberg, 2008.

[85] C. D. Lloyd, "Analysing population characteristics using geographically weighted principal components analysis: A case study of Northern Ireland in 2001," *Computers, Environment and Urban Systems*, vol. 34, no. 5, pp. 389–399, 2010.

[86] D. Hong, F. Yang, J. A. Fessler, and L. Balzano, "Optimally weighted PCA for high-dimensional heteroscedastic data," *SIAM Journal on Mathematics of Data Science*, vol. 5, no. 1, pp. 222–250, 2023.

[87] E. Raponi, H. Wang, M. Bujny, S. Boria, and C. Doerr, "High dimensional Bayesian optimization assisted by principal component analysis," in *Parallel Problem Solving from Nature – PPSN XVI* (T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann, eds.), pp. 169–183, Springer International Publishing, 2020.

[88] K. Antonov, E. Raponi, H. Wang, and C. Doerr, "High dimensional Bayesian optimization with kernel principal component analysis," in *Parallel Problem Solving from Nature – PPSN XVII* (G. Rudolph, A. V. Kononova, H. Aguirre, P. Kerschke, G. Ochoa, and T. Tušar, eds.), pp. 118–131, Springer International Publishing, 2022.

[89] J. Han, M. Kamber, and J. Pei, "3 - Data preprocessing," in *Data Mining (Third Edition)* (J. Han, M. Kamber, and J. Pei, eds.), The Morgan Kaufmann Series in Data Management Systems, pp. 83–124, Boston: Morgan Kaufmann, 3rd ed., 2012.

[90] M. Frean and P. Boyle, "Using Gaussian processes to optimize expensive functions," in *AI 2008: Advances in Artificial Intelligence* (W. Wobcke and M. Zhang, eds.), (Berlin, Heidelberg), pp. 258–267, Springer, Dec 2008.

[91] C. Moore, A. Chua, C. Berry, and J. Gair, "Fast methods for training Gaussian processes on large data sets," *Royal Society Open Science*, vol. 3, May 2016.

[92] Y. Zhang and W. Leithead, "Exploiting Hessian matrix and trust-region algorithm in hyperparameters estimation of Gaussian process," *Applied Mathematics and Computation*, vol. 171, no. 2, pp. 1264–1281, 2005.

[93] S. Geršgorin, "Über die abgrenzung der eigenwerte einer matrix," *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, vol. 6, pp. 749–754, 1931.

[94] L. DeVille, "Optimizing Gershgorin for symmetric matrices," *Linear Algebra and its Applications*, vol. 577, pp. 360–383, 2019.

[95] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific Journal of Mathematics*, vol. 16, pp. 1–3, 1966.

[96] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[97] E. Visser, C. E. van Daalen, and J. Schoeman, "Lossy compression of observations for Gaussian process regression," in *MATEC Web of Conferences*, vol. 370, EDP Sciences, Dec 2022.

[98] K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, USA, 1975.

[99] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[100] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[101] 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.

[102] M. Laguna and R. Martí, "Experimental testing of advanced scatter search designs for global optimization of multimodal functions," *Journal of Global Optimization*, vol. 33, pp. 235–255, Oct 2005.

[103] B. L. Welch, "The generalization of 'Student's' problem when several different population variances are involved," *Biometrika*, vol. 34, pp. 28–35, 01 1947.

[104] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[105] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," Research Report RR-6829, INRIA, 2009.

[106] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009," in *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, (New York, NY, USA), pp. 1689–1696, Association for Computing Machinery, 2010.

[107] W. Huyer and A. Neumaier, "Global optimization by multilevel coordinate search," *Journal of Global Optimization*, vol. 14, pp. 331–355, Jun 1999.

[108] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, pp. 157–181, Oct 1993.

[109] M. J. D. Powell, "The NEWUOA software for unconstrained optimization without derivatives," in *Large-Scale Nonlinear Optimization*, pp. 255–297, Boston, MA, USA: Springer US, 2006.

[110] Z. Pitra, L. Bajer, J. Repický, and M. Holeňa, "Comparison of ordinal and metric Gaussian process regression as surrogate models for CMA evolution strategy," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, (New York, NY, USA), pp. 1764–1771, Association for Computing Machinery, 2017.

[111] W. Huyer and A. Neumaier, "Benchmarking of MCS on the noiseless function testbed." Unpublished manuscript on webpage at `https://arnold-neumaier.at/ms/mcs_exact.pdff`, 2009.

[112] R. Ros, "Benchmarking the NEWUOA on the BBOB-2009 function testbed," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, (New York, NY, USA), pp. 2421–2428, Association for Computing Machinery, 2009.

[113] F. Hutter, H. Hoos, and K. Leyton-Brown, "An evaluation of sequential model-based optimization for expensive blackbox functions," in *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, (New York, NY, USA), pp. 1209–1216, Association for Computing Machinery, 2013.

[114] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, pp. 591–611, Dec 1965.

[115] E. C. Garrido-Merchán and D. Hernández-Lobato, "Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes," *Neurocomputing*, vol. 380, pp. 20–35, Mar 2020.

[116] W. Wang, H.-L. Liu, and K. C. Tan, "A surrogate-assisted differential evolution algorithm for high-dimensional expensive optimization problems," *IEEE Transactions on Cybernetics*, vol. 53, no. 4, pp. 2685–2697, 2023.

[117] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

# Appendix A

# Ideal Transformation Parameter Proofs

This chapter contains the mathematical proofs for the ideal choice of transformation parameters from Equations 9.5 and 9.6 and how these choices ensure that invariant properties (i)–(iv) hold for the set of transformed, observed input points $X'$ and transformed, observed output points $Y'$. The proofs in the rest of this chapter are ordered according to the invariant properties (i.e., first proving that invariant property (i) holds, then proving that invariant property (ii) holds, and so on).

*Proof for invariant property (i).* The transformation defining the mapping between the elements of the set of observed outputs $Y$ and the transformed representation thereof $Y'$ is given as

$$y_i = ay'_i + b \; \forall i \in \{1, \ldots, n\}. \tag{A.1}$$

Rearranging terms yields

$$\therefore y'_i = \frac{y_i - b}{a} \; \forall i \in \{1, \ldots, n\}. \tag{A.2}$$

It must be shown that the choosing the transformation parameters as $a = y_{\max}$ and $b = y_{\min}$ satisfies invariant property (i). In other words, that by using these choices for $a$ and $b$ means that the observed minimum value $y_{\min}$ is mapped to zero and that the observed maximum value $y_{\max}$ is mapped to one. Substituting the choices of $a$ and $b$ into Equation A.1 yields

$$\therefore y'_i = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}} \; \forall i \in \{1, \ldots, n\}. \tag{A.3}$$

The minimum observed output point $y_{\min}$ is defined using

$$\exists y_{\min} \in Y : y_{\min} \leq y_i \; \forall y_i \in Y, \tag{A.4}$$

with a corresponding transformed value $y'_{\min}$ defined using Equation A.2. Similarly, the current maximum observed output value $y_{\max}$ is defined as

$$\exists y_{\max} \in Y : y_{\max} \geq y_i \ \forall y_i \in Y, \tag{A.5}$$

with the corresponding transformed value $y'_{\max}$ also defined using Equation A.2. Substituting this minimum observed value $y_{\min}$ and the corresponding transformed output value $y'_{\min}$ into the transform defined in Equation A.3, or

$$\therefore y'_{\min} = \frac{y_{\min} - y_{\min}}{y_{\max} - y_{\min}} = 0, \tag{A.6}$$

shows that the minimum observed output value is mapped to zero. Substituting the values of $y_{\max}$ and $y'_{\max}$ into Equation A.3, given by

$$\therefore y'_{\max} = \frac{y_{\max} - y_{\min}}{y_{\max} - y_{\min}} = 1, \tag{A.7}$$

shows that the maximum observed output value is mapped to one, concluding the proof. □

*Proof for invariant property (ii).* The transformation defining the mapping between the elements of the set of observed inputs $X$ and the transformed representation thereof $X'$ is given as

$$\mathbf{x}_i = \mathbf{R}\mathbf{S}\mathbf{x}'_i + \mathbf{c} \ \forall i \in \{1, \ldots, n\}. \tag{A.8}$$

It must be shown that the choosing the transformation parameters as $\mathbf{R} = \mathbf{U}$, $\mathbf{S} = \hat{\mathbf{L}}$ and $\mathbf{c} = \mathbf{x}_{\min}$ satisfies invariant property (i). In other words, that for these choices the minimum observed input point $\mathbf{x}_{\min}$ associated with the minimum observed output point $y_{\min}$ is transformed to be at the origin.Substituting these choices of the transformation parameters yields

$$\therefore \mathbf{x}_i = \mathbf{U}\hat{\mathbf{L}}\mathbf{x}'_i + \mathbf{x}_{\min} \ \forall i \in \{1, \ldots, n\}. \tag{A.9}$$

Rearranging terms yields

$$\therefore \mathbf{x}'_i = \hat{\mathbf{L}}^{-1}\mathbf{U}^{\top}(\mathbf{x}_i - \mathbf{x}_{\min}) \ \forall i \in \{1, \ldots, n\}. \tag{A.10}$$

The minimum observed input point $\mathbf{x}_{\min}$ that is associated with the minimum observed output value $y_{\min}$, is defined by

$$\exists(\mathbf{x}_{\min}, y_{\min}) \in \mathcal{D} : y_{\min} \leq y_i \ \forall(\mathbf{x}_i, y_i) \in \mathcal{D}, \tag{A.11}$$

with the corresponding transformed minimum observed input point $\mathbf{x}'_{\min}$ defined using Equation A.10. Substituting the values of $\mathbf{x}_{\min}$ and $\mathbf{x}'_{\min}$ into Eq A.10 yields

$$\begin{aligned} \therefore \mathbf{x}'_{\min} &= \hat{\mathbf{L}}^{-1}\mathbf{U}^\top(\mathbf{x}_{\min} - \mathbf{x}_{\min}) \\ &= \hat{\mathbf{L}}^{-1}\mathbf{U}^\top(\mathbf{0}_n) \\ &= \mathbf{0}_n, \end{aligned} \tag{A.12}$$

which shows that $\mathbf{x}_{\min}$ is transformed to be a vector of zeroes or the origin, concluding the proof. □

*Proof for invariant property (iii).* Suppose that the weighted principal components of the transformed input points $\mathbf{X}'$ is given by the matrix $\hat{\mathbf{U}}$ in the following SVD

$$\mathbf{X}'\mathbf{W} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^\top. \tag{A.13}$$

It must be shown that these weighted principal components are aligned with the coordinate axes, or $\hat{\mathbf{U}} \in \text{diag}(\pm 1, \ldots, \pm 1)$.

The matrix form of Equation A.10, where the $i$-th columns of the matrices $\mathbf{X}$ and $\mathbf{X}'$ correspond to $\mathbf{x}_{\min}$ and $\mathbf{x}'_{\min}$, respectively, can be given by

$$\therefore \mathbf{X}' = \hat{\mathbf{L}}^{-1}\mathbf{U}^\top(\mathbf{X} - \mathbf{x}_{\min}\mathbf{1}_n^\top). \tag{A.14}$$

Substituting this value of $\mathbf{X}'$ into Equation A.13 yields

$$\therefore \hat{\mathbf{L}}^{-1}\mathbf{U}^\top(\mathbf{X} - \mathbf{x}_{\min}\mathbf{1}_n^\top)\mathbf{W} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^\top, \tag{A.15}$$

where the value of $(\mathbf{X} - \mathbf{x}_{\min}\mathbf{1}_n^\top)$ is recognized to be equal to $\mathbf{X}_\odot$ from Equation 7.5. Using this value, this expression can be restated as

$$\therefore \hat{\mathbf{L}}^{-1}\mathbf{U}^\top\mathbf{X}_\odot\mathbf{W} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^\top. \tag{A.16}$$

From Equation 7.7, it can be seen that the SVD of $\mathbf{U}^\top\mathbf{X}_\odot\mathbf{W}$ is given by $\mathbf{I}\mathbf{\Sigma}\mathbf{V}^\top$. Substituting this decomposition for $\mathbf{U}^\top\mathbf{X}_\odot\mathbf{W}$ and noting that the matrix multiplication of $\hat{\mathbf{L}}^{-1}$ and $\mathbf{\Sigma}$ (both of the shape $\mathbb{R}^{d\times d}$) is commutative yields

$$\begin{aligned} \therefore \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^\top &= \hat{\mathbf{L}}^{-1}\mathbf{I}\mathbf{\Sigma}\mathbf{V}^\top \\ &= \mathbf{I}(\hat{\mathbf{L}}^{-1}\mathbf{\Sigma})\mathbf{V}^\top \end{aligned} \tag{A.17}$$

Since $\hat{\mathbf{L}}^{-1}$ and $\mathbf{\Sigma}$ are positive semidefinite diagonal matrices of the same shape, the result must also a positive semidefinite diagonal matrix. This result is therefore a valid SVD of $\mathbf{X}'$

$$\mathbf{X}'\mathbf{W} = \mathbf{I}(\hat{\mathbf{L}}^{-1}\mathbf{\Sigma})\mathbf{V}^\top, \tag{A.18}$$

since $\mathbf{I}$ is orthogonal by definition, $\mathbf{V}^\top$ is orthogonal by virtue of being a result of the SVD of Equation 7.7 and $(\hat{\mathbf{L}}^{-1}\mathbf{\Sigma})$ is a positive semidefinite diagonal matrix as stated previously.

This decomposition implies that the weighted principal components $\hat{\mathbf{U}}$ from Equation A.13 are equal to the identity matrix, concluding the proof. □

*Proof for invariant property (iv).* It must be proven that the most likely length-scales for a GP constructed using the transformed input points $\mathbf{X}'$ and automatic relevance determination along the weighted principal components (described by the rotation matrix $\mathbf{U}$) are unity.

Supposing that rotated automatic relevance determination along non-coordinate axes can be described by the squared exponential kernel

$$k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j \,; \hat{\boldsymbol{\ell}}_{\mathbf{U}}) = \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \hat{\boldsymbol{\Lambda}}_{\mathbf{U}}^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right) + \sigma_n^2 \delta_{ij} \tag{A.19}$$

with the change of basis

$$\begin{aligned} \hat{\boldsymbol{\Lambda}}_{\mathbf{U}}^{-1} &= \mathbf{U}\hat{\boldsymbol{\Lambda}}^{-1}\mathbf{U}^\top \\ \text{where} \quad \hat{\boldsymbol{\Lambda}} &= \text{diag}(\hat{\ell}_1^2, \hat{\ell}_2^2, \ldots, \hat{\ell}_d^2). \end{aligned} \tag{A.20}$$

The diagonal factor $\hat{\boldsymbol{\Lambda}}$ in this kernel can be factorized as

$$\begin{aligned} \hat{\boldsymbol{\Lambda}}_{\mathbf{U}}^{-1} &= \mathbf{U}(\hat{\mathbf{L}}\,\hat{\mathbf{L}})^{-1}\mathbf{U}^\top \\ &= \mathbf{U}\hat{\mathbf{L}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{U}^\top \\ \text{where} \quad \hat{\mathbf{L}} &= \text{diag}(\hat{\ell}_1, \hat{\ell}_2, \ldots, \hat{\ell}_d) \end{aligned} \tag{A.21}$$

Substituting the values of $\mathbf{x}_i$ and $\mathbf{x}_j$ from Equation A.9 into the kernel defined in Equation A.19 and noting that $\mathbf{L}$ is orthogonal ($L^\top = L$), simplifying yields

$$\begin{aligned} \therefore k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j \,; \hat{\boldsymbol{\ell}}_{\mathbf{U}}) &= \sigma_f^2 \cdot \exp\Big(-\frac{1}{2}(\mathbf{U}\hat{\mathbf{L}}\mathbf{x}_i' - \mathbf{x}_{\min} - \mathbf{U}\hat{\mathbf{L}}\mathbf{x}_j' + \mathbf{x}_{\min})^\top \\ &\quad \times \hat{\boldsymbol{\Lambda}}_{\mathbf{U}}^{-1}(\mathbf{U}\hat{\mathbf{L}}\mathbf{x}_i' - \mathbf{x}_{\min} - \mathbf{U}\hat{\mathbf{L}}\mathbf{x}_j' + \mathbf{x}_{\min})\Big) + \sigma_n^2\delta_{ij} \\ &= \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{U}\hat{\mathbf{L}}\mathbf{x}_i' - \mathbf{U}\hat{\mathbf{L}}\mathbf{x}_j')^\top \hat{\boldsymbol{\Lambda}}_{\mathbf{U}}^{-1}(\mathbf{U}\hat{\mathbf{L}}\mathbf{x}_i' - \mathbf{U}\hat{\mathbf{L}}\mathbf{x}_j')\right) + \sigma_n^2\delta_{ij} \\ &= \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}_i' - \mathbf{x}_j')^\top \hat{\mathbf{L}}\mathbf{U}^\top \hat{\boldsymbol{\Lambda}}_{\mathbf{U}}^{-1}\mathbf{U}\hat{\mathbf{L}}(\mathbf{x}_i' - \mathbf{x}_j')\right) + \sigma_n^2\delta_{ij} \end{aligned} \tag{A.22}$$

Substituting the value of $\hat{\boldsymbol{\Lambda}}_{\mathbf{U}}^{-1}$ for the value from Equation A.20, noting that the matrix $\mathbf{U}$ is orthogonal (i.e, $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$), simplifying yields

$$\begin{aligned}
\therefore k_{\text{SE}}(\mathbf{x}_i, \mathbf{x}_j\,; \hat{\boldsymbol{\ell}}_{\mathbf{U}}) &= \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}'_i - \mathbf{x}'_j)^\top \hat{\mathbf{L}}\mathbf{U}^\top \mathbf{U}\hat{\mathbf{L}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{U}^\top \mathbf{U}\hat{\mathbf{L}}(\mathbf{x}'_i - \mathbf{x}'_j)\right) + \sigma_n^2\delta_{ij} \\
&= \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}'_i - \mathbf{x}'_j)^\top \hat{\mathbf{L}}\hat{\mathbf{L}}^{-1}\hat{\mathbf{L}}^{-1}\hat{\mathbf{L}}(\mathbf{x}'_i - \mathbf{x}'_j)\right) + \sigma_n^2\delta_{ij} \\
&= \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}'_i - \mathbf{x}'_j)^\top \mathbf{I}\,(\mathbf{x}'_i - \mathbf{x}'_j)\right) + \sigma_n^2\delta_{ij} \\
&= k_{\text{SE}}(\mathbf{x}'_i, \mathbf{x}'_j\,; \mathbf{1}_n).
\end{aligned} \tag{A.23}$$

Therefore, constructing a GP using the transformed input points $X'$ using unit length scales along the coordinate axes would be equivalent to constructing a GP using the untransformed inputs $X$ with length-scales along the weighted principal components $\mathbf{U}$, concluding the proof. □

# Appendix B

# Full Synthetic Test Function Benchmark Results

The full results of the comparative algorithm study using synthetic test functions from Section 10.1 is provided in this chapter. Each of the compared algorithms chosen in Section 10.1 are applied to each test function, defined in Table 10.1, for 50 independent runs. In Table B.1 and B.2, the means and standard deviations of the minimum global regret ($y_{\min} - f_{\min}$) reached and the total wall-clock times needed for each run (in effect, for 150 objective function samples while excluding the time needed to calculate the objective function for each sample) are reported.

Similarly to the analysis used by Xie et al. [49] and Wang et al. [116], for each test function, a Wilcoxon rank-sum test [117] (also known as the Mann-Whitney U test) is performed for the minimum global regret for each algorithm compared to the LABCAT algorithm. We indicate the results of this test using "+", "$\approx$" and "$-$", if the LABCAT algorithm performs statistically significantly better than, comparable to, or worse than the compared algorithm, respectively. The result is considered to be statistically significant if the $p$-value is less than 0.05, adjusted using a Bonferroni correction [104] to $\frac{0.05}{8} = 0.00625$ using the number of algorithms compared against LABCAT.

| | Sphere | | | Quartic | | | Booth | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | $\mu \pm (\sigma)$ | $+/\approx/-$ | Time (s) | $\mu \pm (\sigma)$ | $+/\approx/-$ | Time (s) | $\mu \pm (\sigma)$ | $+/\approx/-$ | Time (s) |
| LABCAT | 5.68e−17 ± (7.44e−17) | N/A | 0.055 ± (0.011) | 2.79e−22 ± (6.40e−22) | N/A | 0.055 ± (0.007) | 9.98e−16 ± (1.28e−15) | N/A | 0.055 ± (0.012) |
| BADS | 3.80e−09 ± (1.20e−08) | + | 5.448 ± (0.704) | 1.20e−08 ± (5.29e−08) | + | 4.890 ± (0.758) | 4.88e−06 ± (2.25e−05) | + | 6.868 ± (0.801) |
| BO | 1.36e−05 ± (1.93e−05) | + | 21.507 ± (0.779) | 1.52e−08 ± (1.81e−08) | + | 24.213 ± (0.933) | 4.87e−05 ± (5.48e−05) | + | 24.271 ± (1.119) |
| Random | 1.32e−01 ± (1.18e−01) | + | 0.010 ± (0.005) | 5.38e−04 ± (1.15e−03) | + | 0.009 ± (0.002) | 2.62e+00 ± (1.92e+00) | + | 0.010 ± (0.005) |
| SRSM | 2.14e−02 ± (1.50e−01) | + | 34.696 ± (2.249) | 9.17e−10 ± (5.90e−09) | + | 29.548 ± (0.916) | 5.77e−05 ± (7.77e−05) | + | 37.755 ± (1.628) |
| TREGO | 1.73e−03 ± (2.80e−03) | + | 111.496 ± (2.323) | 3.82e−04 ± (1.91e−03) | + | 111.667 ± (8.665) | 6.23e−02 ± (1.30e−01) | + | 109.807 ± (1.846) |
| TRLBO | 1.34e−04 ± (3.37e−04) | + | 1.408 ± (0.067) | 1.90e−04 ± (5.24e−04) | + | 1.427 ± (0.110) | 2.64e−02 ± (5.75e−02) | + | 1.326 ± (0.073) |
| TuRBO-1 | 3.36e−04 ± (3.18e−04) | + | 1.407 ± (0.124) | 8.94e−09 ± (5.47e−08) | + | 1.474 ± (0.102) | 4.67e−03 ± (1.13e−02) | + | 1.446 ± (0.134) |
| TuRBO-5 | 1.22e−03 ± (1.27e−03) | + | 4.019 ± (0.442) | 4.40e−06 ± (6.09e−06) | + | 4.138 ± (0.339) | 3.36e−02 ± (3.63e−02) | + | 4.005 ± (0.392) |

Table B.1: Average and standard deviation of the minimum global regret, their statistical comparisons according to a rank-sum test, and mean and standard deviation of the wall-clock times for 50 independent runs on synthetic test functions $f_1$ to $f_3$.

| | Rosenbrock | | | Branin-Hoo | | | Levy | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | $\mu \pm (\sigma)$ | $+/\approx/-$ | Time (s) | $\mu \pm (\sigma)$ | $+/\approx/-$ | Time (s) | $\mu \pm (\sigma)$ | $+/\approx/-$ | Time (s) |
| LABCAT | 1.08e−10 ± (1.36e−10) | N/A | 0.056 ± (0.011) | 1.71e−11 ± (3.02e−11) | N/A | 0.091 ± (0.046) | 1.26e−01 ± (5.95e−01) | N/A | 0.061 ± (0.018) |
| BADS | 6.86e−04 ± (9.92e−04) | + | 6.807 ± (0.647) | 4.57e−07 ± (1.27e−06) | + | 5.182 ± (0.795) | 4.25e−07 ± (1.76e−06) | + | 4.983 ± (0.861) |
| BO | 2.76e−01 ± (3.22e−01) | + | 35.238 ± (1.950) | 3.08e−05 ± (2.71e−05) | + | 25.093 ± (1.075) | 3.47e−05 ± (3.67e−05) | + | 24.546 ± (1.232) |
| Random | 4.04e+00 ± (4.85e+00) | + | 0.017 ± (0.004) | 4.14e−01 ± (4.12e−01) | + | 0.010 ± (0.003) | 2.01e−01 ± (2.02e−01) | + | 0.010 ± (0.004) |
| SRSM | 3.43e+00 ± (4.12e+00) | + | 32.168 ± (2.477) | 2.39e−05 ± (3.93e−05) | + | 36.125 ± (1.802) | 2.40e−05 ± (1.02e−04) | + | 35.992 ± (2.253) |
| TREGO | 1.25e+00 ± (1.33e+00) | + | 110.343 ± (2.684) | 4.50e−03 ± (1.29e−02) | + | 110.813 ± (2.427) | 2.83e−03 ± (3.46e−03) | + | 110.802 ± (2.280) |
| TRLBO | 1.68e+00 ± (3.05e+00) | + | 1.318 ± (0.062) | 8.52e−03 ± (4.03e−02) | + | 1.407 ± (0.077) | 2.13e−01 ± (6.32e−01) | + | 1.393 ± (0.057) |
| TuRBO-1 | 2.35e+00 ± (3.27e+00) | + | 1.417 ± (0.120) | 1.58e−03 ± (2.23e−03) | + | 1.445 ± (0.137) | 1.05e−03 ± (3.97e−03) | + | 1.499 ± (0.204) |
| TuRBO-5 | 1.65e+00 ± (1.46e+00) | + | 4.144 ± (0.254) | 9.21e−03 ± (1.11e−02) | + | 3.983 ± (0.353) | 9.13e−03 ± (1.61e−02) | + | 3.328 ± (0.536) |

Table B.2: Average and standard deviation of the minimum global regret, their statistical comparisons according to a rank-sum test, and mean and standard deviation of the wall-clock times for 50 independent runs on synthetic test functions $f_4$ to $f_6$.

# Appendix C

# COCO Benchmark Results

## C.1 LABCAT Ablation Study COCO Results

### C.1.1 Primary Ablation Study Results

The full results of the primary LABCAT ablation study from Section 10.3 using the COCO benchmark are provided in this section.

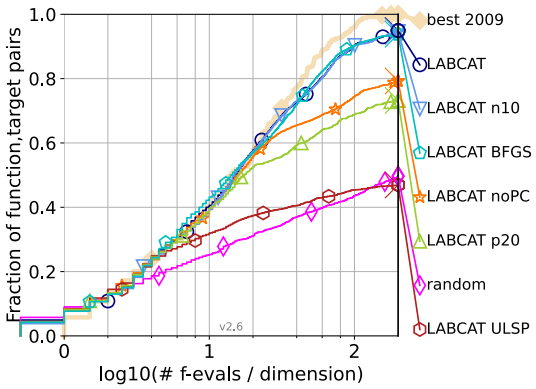
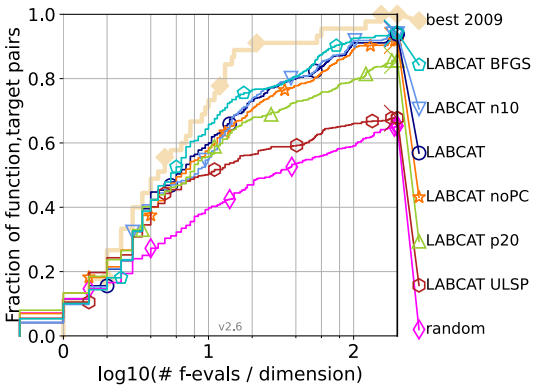
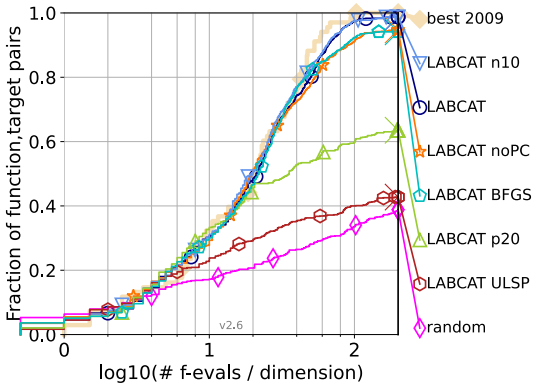
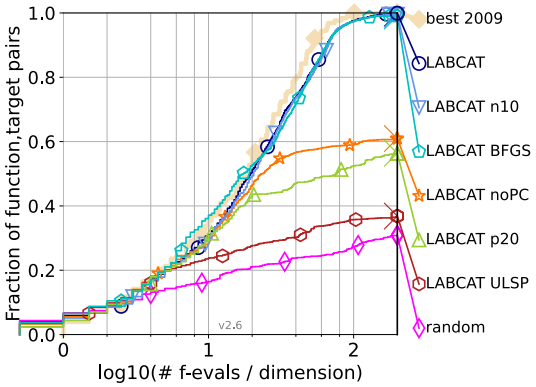
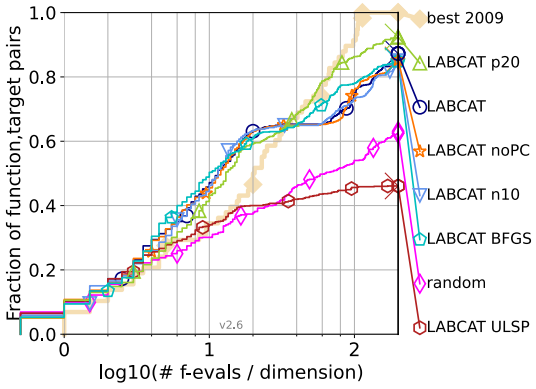
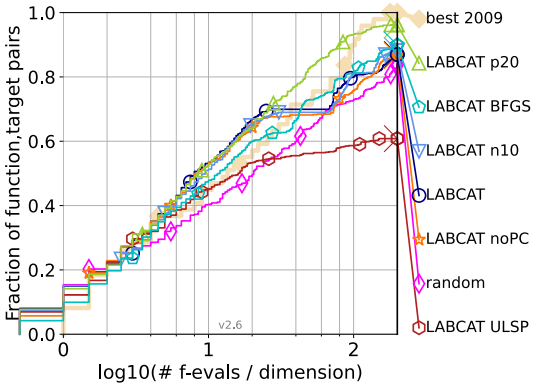
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.1: 2-D runtime ablation ECDFs table from the COCO benchmark.

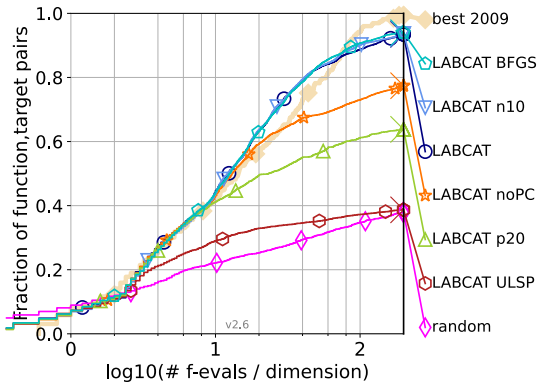
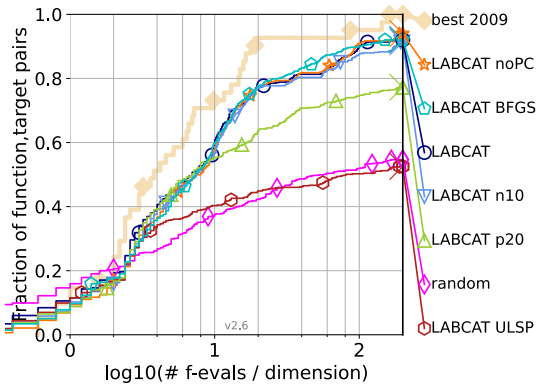
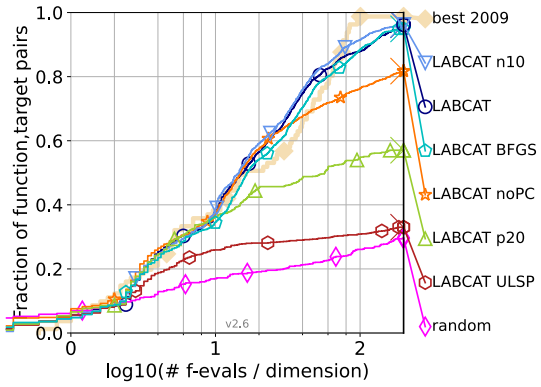
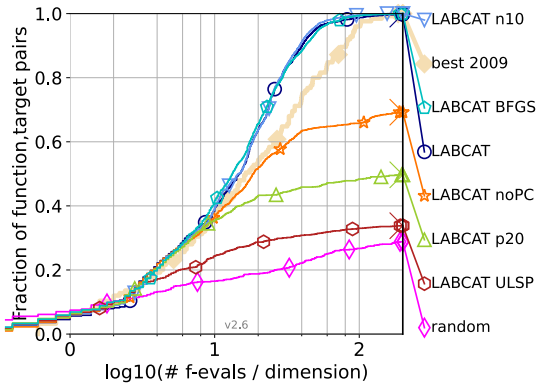
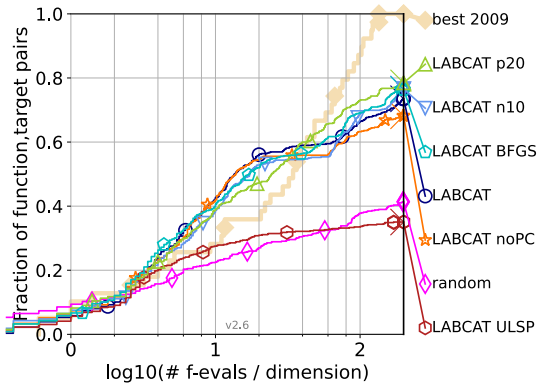
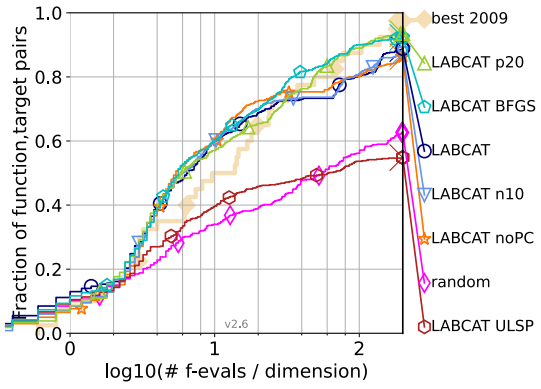
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.2: 5-D runtime ablation ECDFs table from the COCO benchmark.

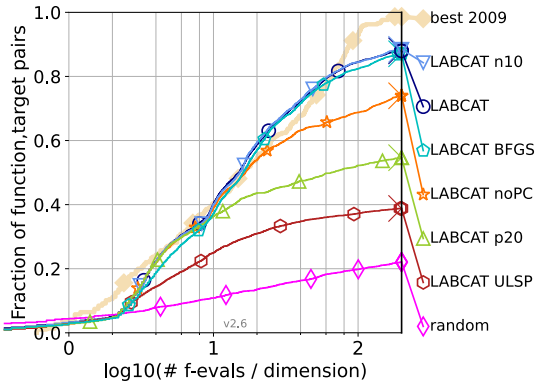
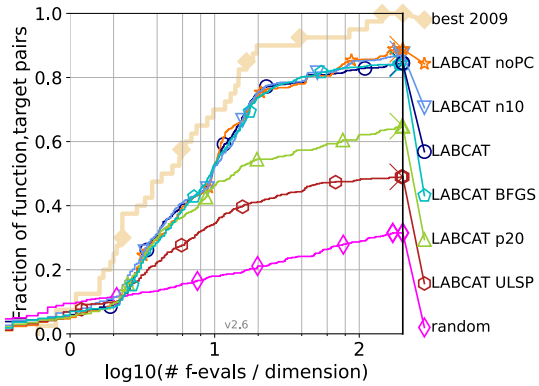
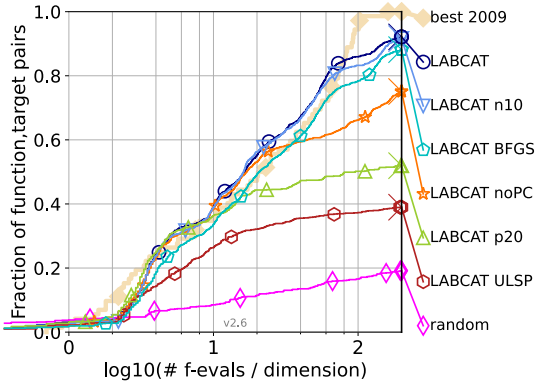
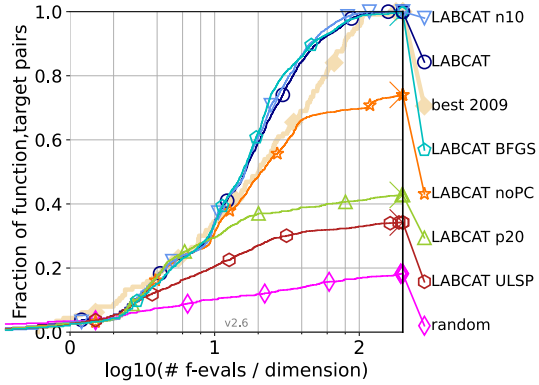
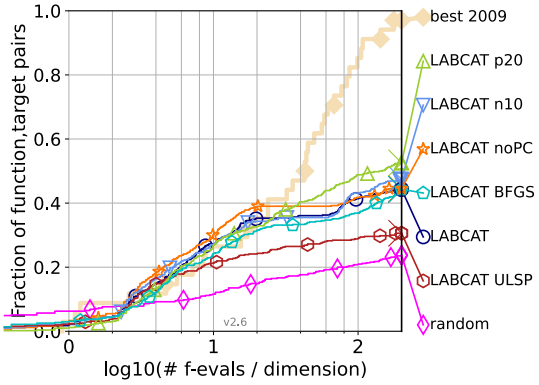
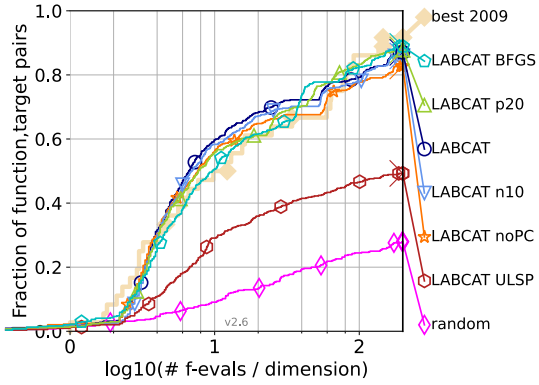
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.3: 10-D runtime ablation ECDFs table from the COCO benchmark.

### C.1.2 Secondary Ablation Study Results

The full results of the secondary LABCAT ablation study from Section 10.3 using the COCO benchmark are provided in this section.

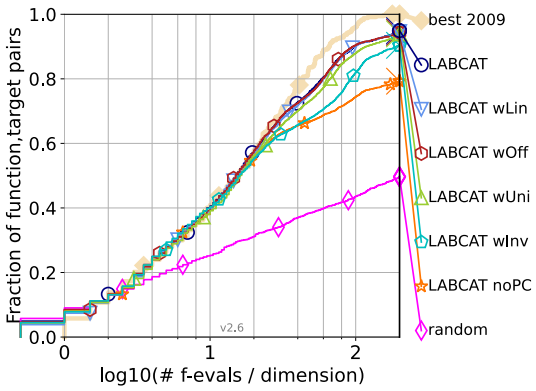
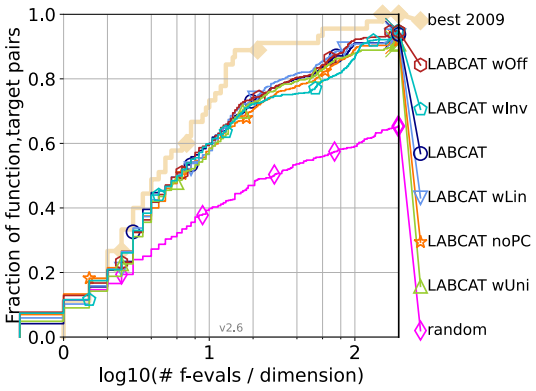
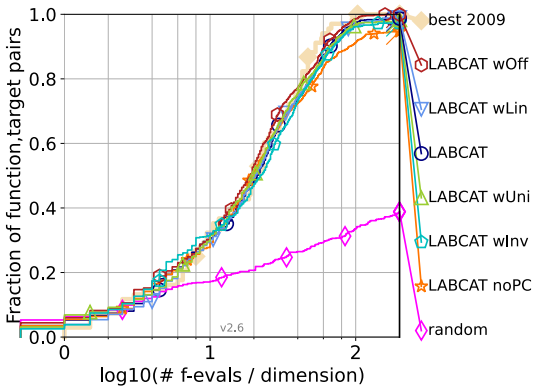
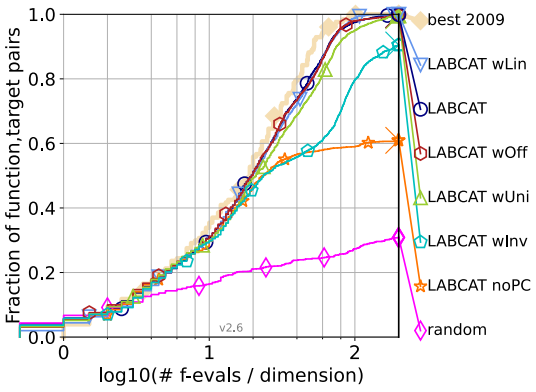
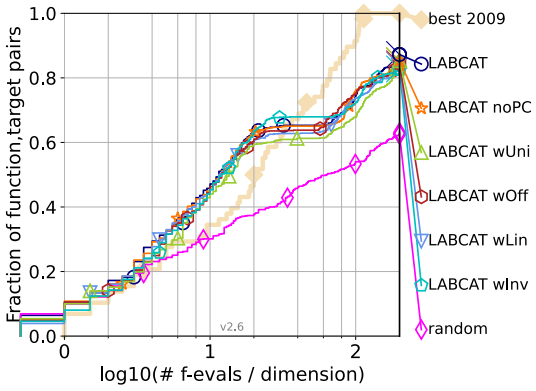
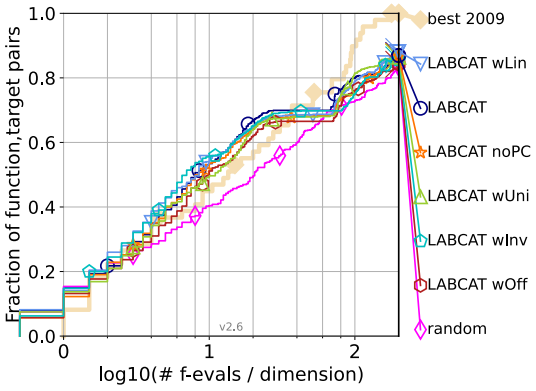
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.4: 2-D runtime secondary ablation ECDFs table from the COCO benchmark.

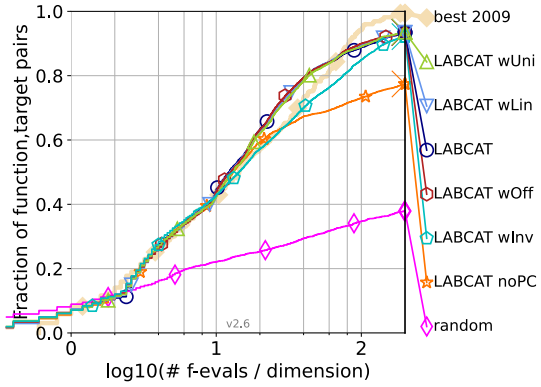
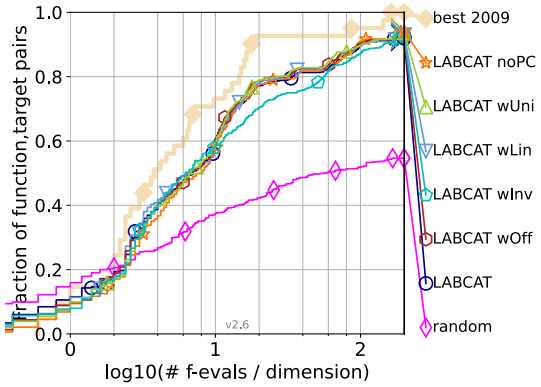
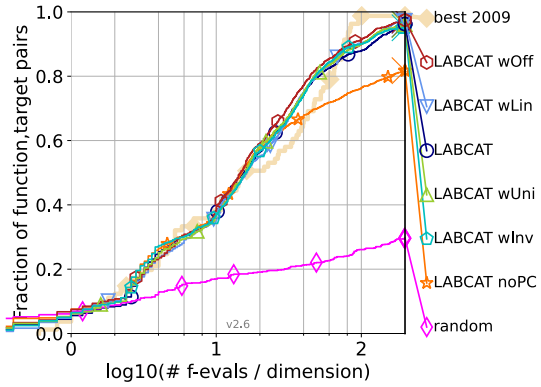
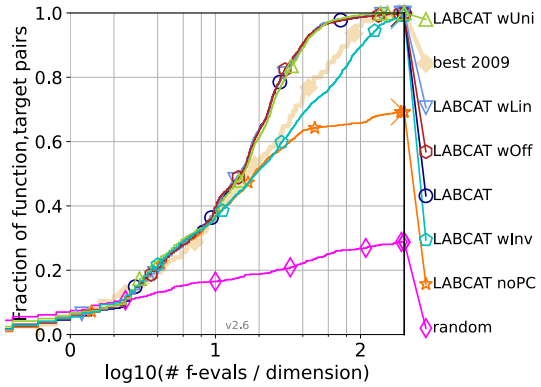
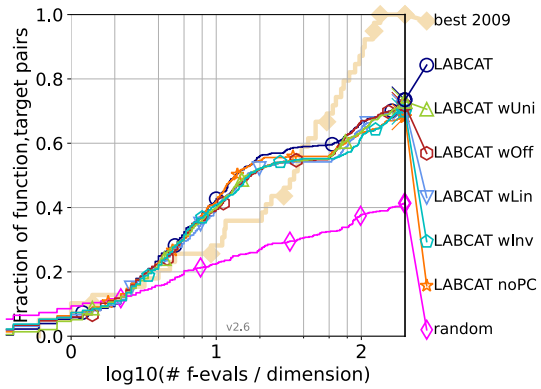
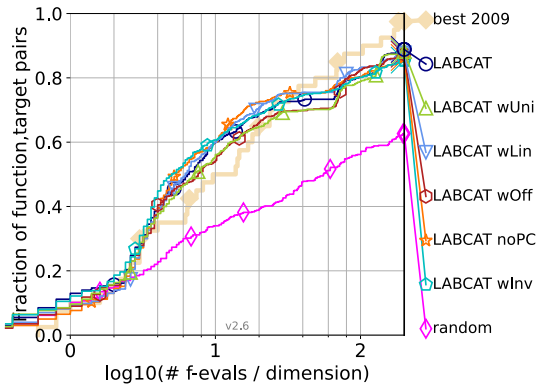
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.5: 5-D runtime secondary ablation ECDFs table from the COCO benchmark.

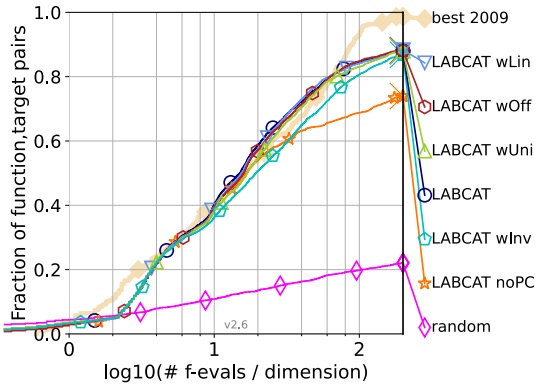
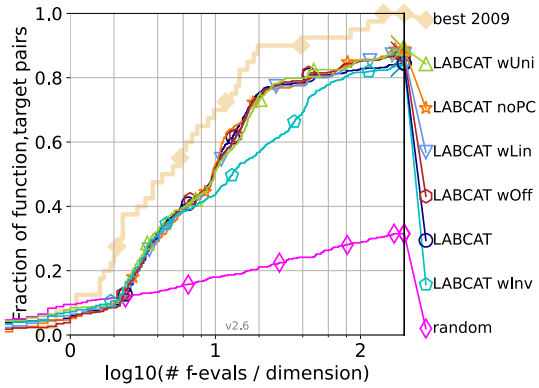
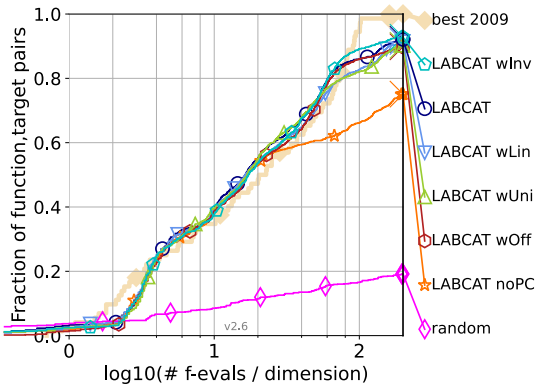
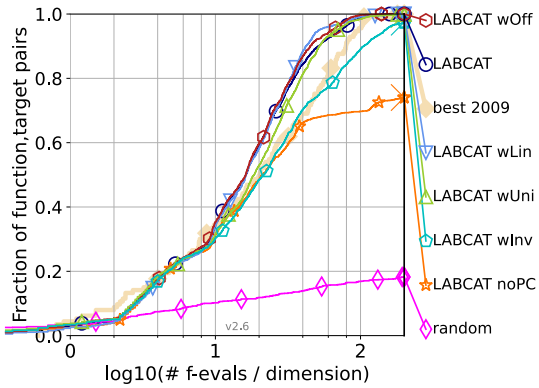
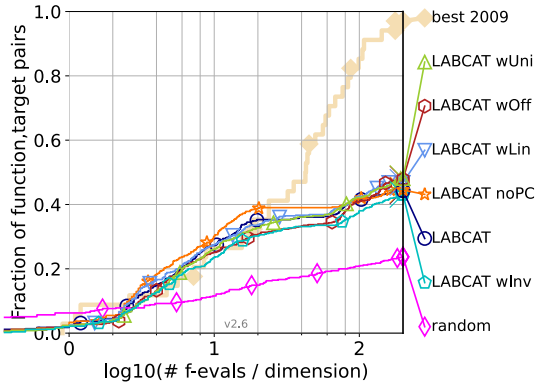
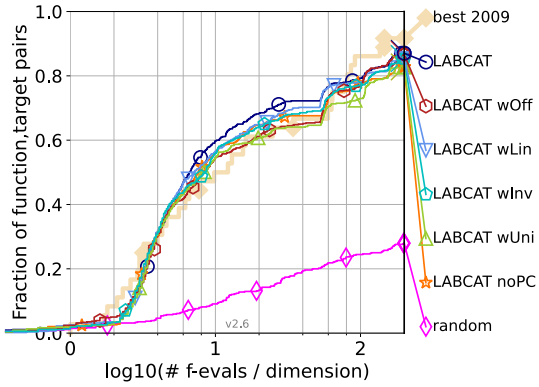
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.6: 10-D runtime secondary ablation ECDFs table from the COCO benchmark.

## C.2 Full COCO Comparative Study Results

The full results of the comparative algorithm study from Section 10.4 using the COCO benchmark are provided in this section.

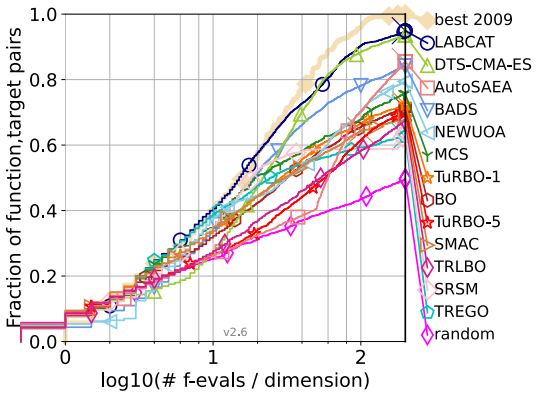
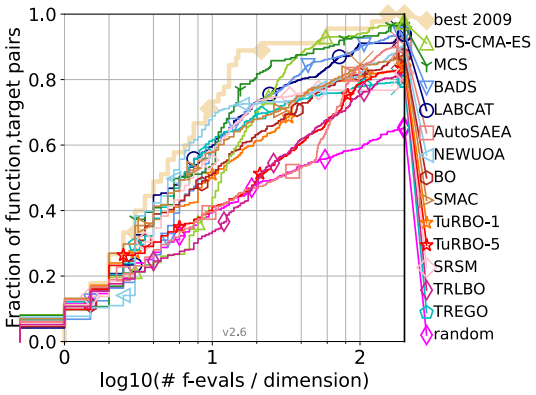
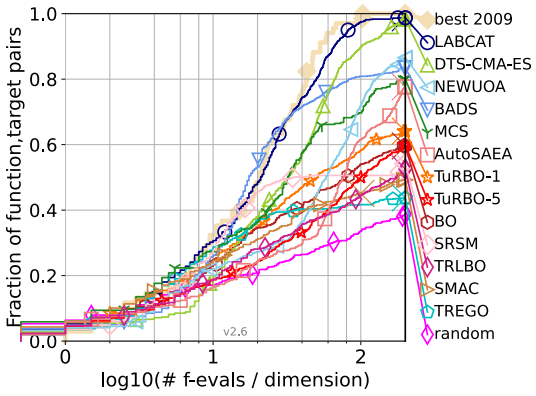
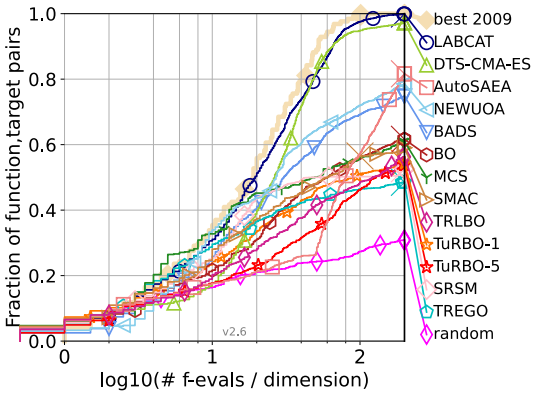
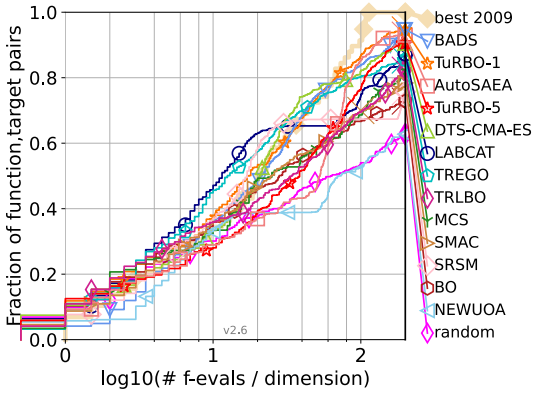
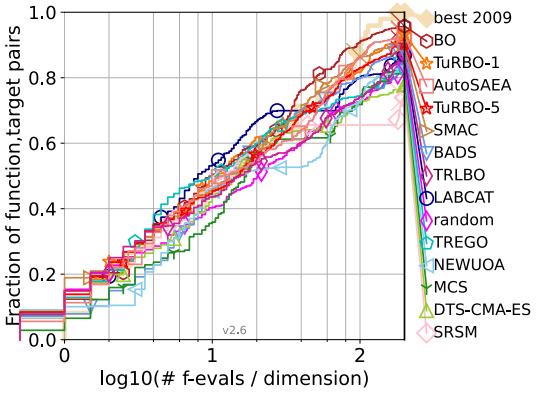
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.7: 2-D comparative study runtime ECDFs table from the COCO benchmark.

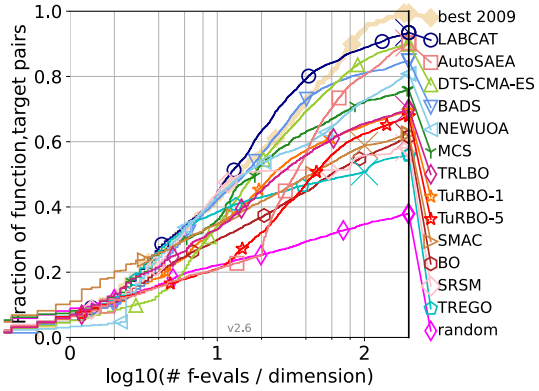
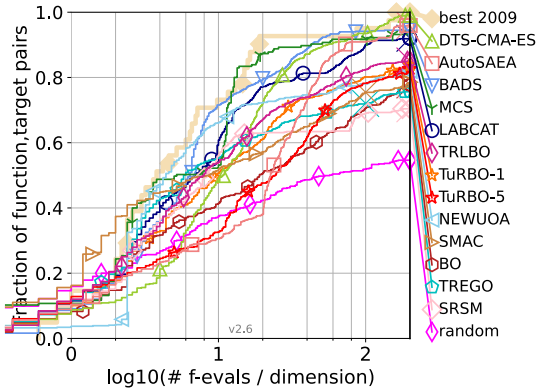
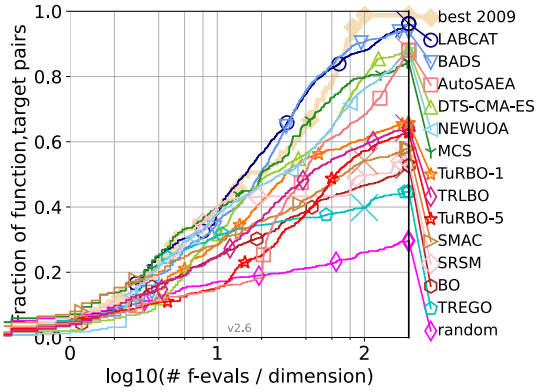
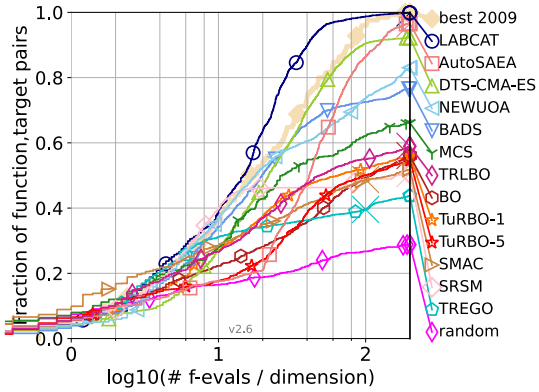
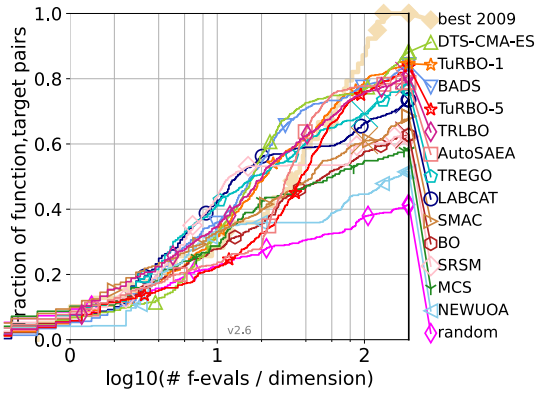
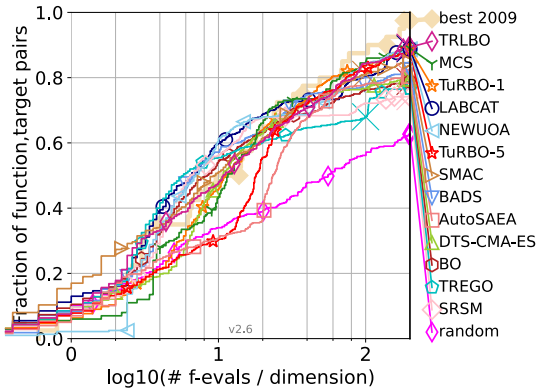
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.8: 5-D comparative study runtime ECDFs table from the COCO benchmark.

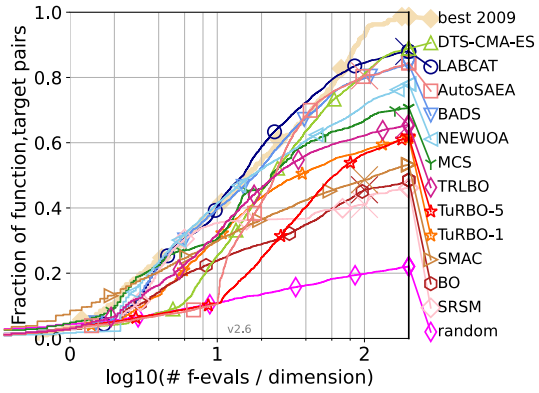
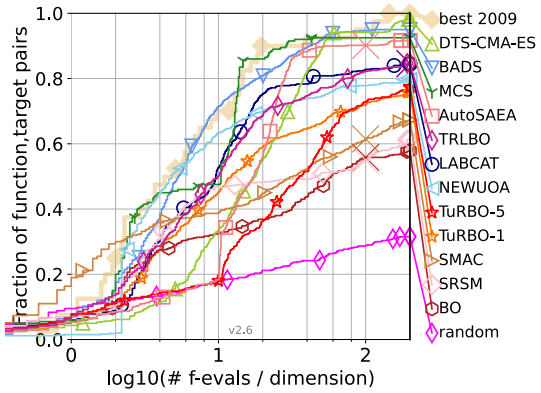
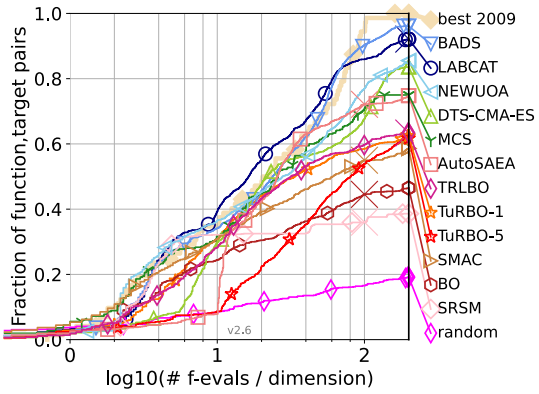
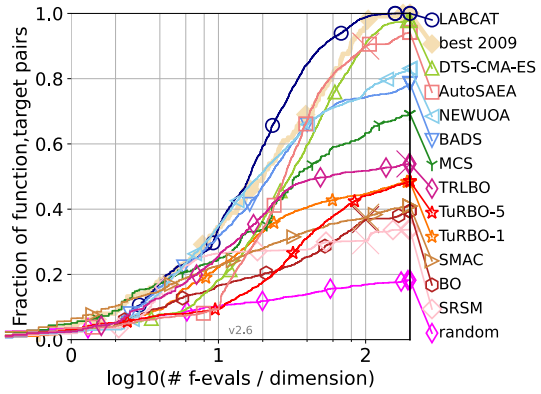
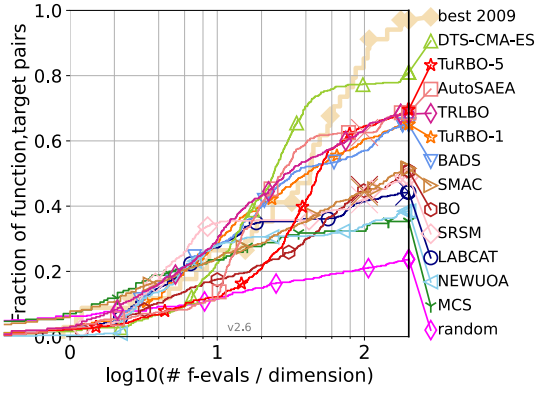
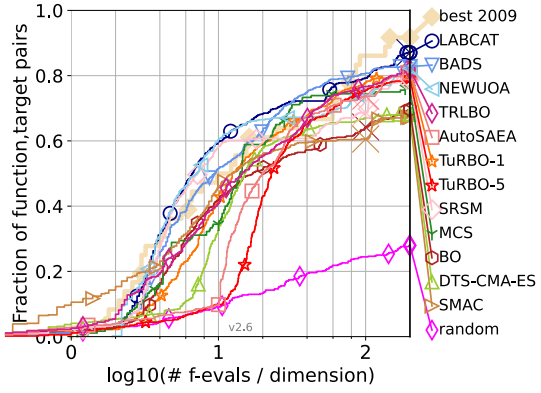
| All Functions | (i) Separable |
|---|---|
|  |  |
| (ii) Unimodal, Low Conditioning | (iii) Unimodal, High Conditioning |
|  |  |
| (iv) Multimodal, Adequate Structure | (v) Multimodal, Weak Structure |
|  |  |



Table C.9: 10-D comparative study runtime ECDFs table from the COCO benchmark.