Institute of Automatic Control

RWTH AACHEN UNIVERSITY

The present work was submitted to the Institute of Automatic Control of RWTH Aachen University

Black-box Optimization Methods for Real-time Parameter Tuning in Robotics

Master's thesis
of
Ivan Wilberg Martins de Oliveira, B. Sc.
Matriculation Number 420972

Field of study:	Automation Technology
Duration of this work:	24 Weeks
Submitted on:	18.05.2026
Number:	S2372
Supervisor:	Sebastian Stemmler, Dr.-Ing
External supervisor:	Johannes Englsberger, Dr.-Ing

Contents

List of Acronyms

ARD Automatic Relevance Determination

BID Biologically Inspired Dead-Beat (controller)

BO Bayesian Optimization

CMA-ES Covariance Matrix Adaptation Evolution Strategy

CoM Center of Mass

DoF Degrees of Freedom

EDA Estimation of Distribution Algorithm

ELBO Evidence Lower Bound

ES Evolution Strategy

GP Gaussian Process

GP-CMA-ES Gaussian Process-CMA-ES

GPML Gaussian Processes for Machine Learning

KL Kullback Leibler divergence

LCB Lower Confidence Bound

LLC Low Level Controller

LRA Learning Rate Adaptation

MRF-EDA Mahalanobis, Resampling, Fitness-based Estimation of Distribution Algorithm

NM Nelder-Mead

PID Proportional-Integral-Derivative

QP Quadratic Program

SA Simulated Annealing

SVM Support Vector Machine

VI Variational Inference

WBC Whole-Body Controller

List of Symbols

1_n Vector of ones in $\mathbb{R}^n$

L Constant in finite-difference error bound (e.g., smoothness/Lipschitz-related)

ℓ Kernel length scale (Matern $5/2$)

$\epsilon(x;\eta)$ Noise/error term in finite-difference gradient estimate

$\|\cdot\|_\infty$ Infinity norm

λ Population size

$\mathbb{R}^n$ n-dimensional real vector space (search space)

$\mathbb{R}$ Set of real numbers

$\mathbf{C}$ Covariance matrix of the sampling distribution

$\mathbf{I}_n$ $n \times n$ identity matrix

$\mathbf{K}_*$ Predictive covariance (or variance) of the GP at $\mathbf{x}_*$

$\mathbf{K}_{\mathbf{xx}}$ Kernel matrix over training inputs $\mathbf{x}$

$\mathbf{K}$ GP kernel matrix

$\mathbf{L}$ Cholesky factor of $\mathbf{C}$

$\mathbf{Q}_{\mathbf{xx}}$ Inducing-point covariance approximation term in SVGP

$\mathbf{k}_*$ Kernel vector between $\mathbf{x}_*$ and training inputs

$\mathbf{m}_*$ Predictive mean of the GP at $\mathbf{x}_*$

$\mathbf{m}$ Mean vector of the sampling distribution

$\mathbf{p}_\sigma$ Evolution path for step-size control

$\mathbf{p}_c$ Evolution path for covariance matrix adaptation

$\mathbf{r}$ Distance term in kernel definition

$\mathbf{u}_{k:\lambda}$ Normalized step of the k-th ranked individual

$\mathbf{x}_*$ Test input location for GP prediction

$\mathbf{x}_k$ k-th sampled individual

$\mathbf{x}_u$ Inducing inputs / inducing points for sparse variational GP

$\mathbf{x}_{k:\lambda}$ k-th best individual among λ candidates (ranked by increasing cost)

$\mathbf{x}$ Candidate solution / individual

$\mathbf{y}$ Cost / objective function values of sampled individuals

$\mathbf{z}$ Whitened sample vector

$\mathcal{D}$ Generative sampling distribution

$\mathcal{L}(q)$ Evidence lower bound

μ_{eff} Effective selection mass

μ Number parents

ρ Kernel output variance

σ_n^2 Observation noise variance

σ Step size in CMA-ES

θ Parameters

c_1 Learning rate for rank-one covariance update term

c_μ Learning rate for rank-μ covariance update term

c_σ Learning rate for the step-size evolution path update

c_c Learning rate for the covariance evolution path update

d_σ Damping factor

$f(\mathbf{x})$ Latent function modeled by the Gaussian process

f_* Latent GP prediction at $\mathbf{x}_*$

$g_{\max}$ Maximum number of generations

g Generation index

h_σ Heaviside indicator

j Number of training data points used in the dense GP

$k(\mathbf{x}, \mathbf{x}')$ Kernel function

$m(\mathbf{x})$ GP mean function

n Dimension of the search space

$q(\theta)$ Variational approximation distribution over parameters

u Number of inducing points in sparse GP approximation

w_k Recombination weight of the k-th best-ranked solution

Abstract

Hyperparameter tuning is essential for the optimal functioning of robotic systems. Currently, this task is often performed by experts using a combination of heuristics and random search, a time-consuming, hard-to-replicate process. Finding appropriate hyperparameters is an even more pronounced problem on robots with a high number of degrees of freedom, such as humanoids. This work proposes to automate tuning procedures using black box optimization, in particular the CMA-ES algorithm. Adaptations are proposed to the standard form of CMA-ES to increase performance on tasks relevantly similar to the use case. A Gaussian process surrogate assisted version of CMA-ES with an epistemic-uncertainty-based step-size modulation mechanism is proposed. The real world use case consists of tuning a humanoid robot to perform a running task with minimum energy expenditure. Algorithms are first tested on canonical black box optimization benchmarking functions and subsequently in simulation. The developed algorithms resulted in the finding of hyperparameters yielding up to a 10% lower cost than that obtained using hyperparameters found by hand tuning in the running task. These results show that black box optimization through CMA-ES variants is a viable research avenue for hyperparameter tuning in 30-dimensional spaces.

1 Introduction

The performance of a robotic system is greatly influenced by hyperparameters. These can have intuitive effects on the system at hand, but often, when working with more complex systems, precise effects become harder to predict. Expert operators and researchers often tune these systems based on past experiences, a process that is hard to substantiate, scale, or replicate. Moreover, it is possible that large regions of the space of possible hyperparameters remain unexplored, and more effective combinations are not considered in the tuning process. It is an active research topic in robotics to explore ways of partly automating this task. Methods seeking to leverage computational power over hand-crafted heuristics have proven effective in automating the performance of search tasks in the past, at times yielding better results than methods that rely on hand-crafted heuristics [1],[2]. General-purpose black box optimization methods are used to tackle the problem of tuning the hyperparameters of a humanoid robot performing a running task with the minimum possible energy expenditure. Particular focus is given to the state of the art covariance matrix adaptation - evolution strategy (CMA-ES) [3] and possible modifications to its core working principle, all in the interest of discovering acceptable hyperparameters with the lowest possible number of simulation runs. Benchmarking of performance is done on canonical test functions used in the field of black box optimization and on a simulation of a running robot, paving the way for safe real-time tuning on real robot hardware.

2 State of the art

2.1 Hyperparameter tuning

Hyperparameter tuning involves the identification of optimal values for the minimization of one or more cost functions derived from a real system. Hyperparameters differ from parameters in that they are set before deployment of the system in question and are not changed during its operation [4]. It is a task that often shows up in machine learning, robotics and even optimization algorithms themselves [5], [6], [7]. In robotics, two prominent use cases consist of manipulation [8] and gait optimization [9], where the parameters in question range from policy network weights, reward function weights, or reference trajectories. In gait optimization specifically, the number of parameters can vary between 4 [9] to upwards of 30 [10]. They include penalty weights for torque commands in optimal controllers, transition thresholds between gait cycle phases for the state machines of the higher level controllers, desired reference angles of certain joints, among others [10],[11]. Different metrics can be optimized, chief among them being the velocity of the center of mass (CoM) or overall energy consumption of the robot [9]. In the current paradigm, commands are given in the force and torque domain in order to follow reference velocity and position trajectories, avoiding areas that would violate constraints imposed on the latter quantities. This task can be adequately performed using proportional-integral-derivative (PID) control [12]. Evaluating optimal hyperparameters is often a costly endeavor, usually involving many computationally expensive experiments in real hardware or in simulation. It is then crucial that the number of evaluations of the cost function by those means be kept to a minimum [13]. State-of-the-art in hyperparameter tuning often involves starting points and heuristics derived from expert knowledge, rendering experiments ad hoc and difficult to reproduce [14]. This expertise is often used to guarantee feasibility on real hardware by observing safety and feasibility constraints. It often makes use of the fact that many real-world problems have cost functions that are predominantly sensitive to only a subset of the system's hyperparameters [14]. In order to automate this task, several procedures have been proposed, the chief commonality of many being the treatment of the optimization as a black box problem due to the unavailability of gradient information [13].

2.2 Black box optimization

Black box optimization is defined as the task to minimize a cost function in $\mathbb{R}^n \to \mathbb{R}$ by choosing the correct n-dimensional vector that minimizes a scalar output, without access to the analytic function form or gradient with respect to the loss [15]. Due to the unavailability of such information, the optimization of such problems can take two broad approaches. Either one must use brute force and rely on a high number of evaluations or, if due to a limited pool of available function evaluations, estimate a search direction for further queries using a heuristic [15]. Inside the family of classical heuristic algorithms, the following four groups stand out [13]:

1. Simple search heuristics like the Nelder-Mead (NM) method and the Simulated Annealing (SA) method, which look at the locations of existing individual solutions to propose new ones. NM takes a deterministic approach, creating and modifying a simplex with the evaluated cost function values at its vertices. The method is, however, extremely sensitive to noise and fails in high-dimensional problems [16]. SA takes a stochastic approach, proposing solutions in the vicinity of the current best candidate, accepting better performing solutions always and worst ones based

on a selection criterion [17]. SA has been shown to work in high dimensions, but needs problem-specific tuning of its own hyperparameters to perform reliably on each specific high-dimensional problem [18].

2. Bayesian Optimization (BO) [19] which tend to excel in low dimensions. These methods train a surrogate of the cost function and query it in areas where it predicts low function values. These methods have been extended in such ways as to be able to compress information from higher-dimensional datasets into relevant, lower-dimensional latent spaces in order to enable high-dimensional optimization [20]. This, however, comes at the cost of accuracy, as information is lost in this compression into latent components [21]. Other research directions have sought to localize the search in order to better allow for the modeling of local features of the cost function [22].

3. Estimation of distribution (EDA) derivatives [23], a subclass of evolutionary strategy (ES) algorithms, the most refined form of which being the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [24]. These algorithms derive statistics used for updates from a set of solutions, condensing past information into a set of parameters. These algorithms sacrifice some sample efficiency in order to achieve excellent performance in very high dimensional spaces ($\mathbb{R}^n = 50, \ldots, 500$) [3]. Some work has gone into increasing CMA-ES style algorithms' sample efficiency by reducing the number of function evaluations performed [25] or by fitting a cheaper-to-evaluate surrogate model of the cost function to avoid querying the real cost function [26],[27].

4. Particle swarm optimization algorithms create a population of candidate solutions and use global information about the best solution and local information from each candidate solution to update their positions. This algorithm class, however, also struggles in high dimensions when not combined with other heuristics that compensate for their shortcomings [28].

For the use case of hyperparameter optimization using black box algorithms in robotics, the number of hyperparameters one expects to encounter can be relatively high. It is important that the chosen algorithm be able to cope with this high dimensionality [10] and that the algorithm be as general-purpose as possible, requiring little expert tuning when moving between different kinds of robots or tasks [6]. Changes to sampling, parameter updates, and preselection before evaluation have often been shown to increase performance in test functions [29], and the avenue of surrogate-assisted CMA-ES variants is still under early exploration [27].

2.3 Research objectives

For the reasons mentioned at the end of section 2.2, the EDA and CMA-ES have been chosen as the preferred algorithms to be used in this work. This work seeks to build upon the body of literature that modifies the EDA and CMA-ES to shore up their weakness of sample inefficiency. Of particular interest would be the investigation of the performance of CMA-ES derivatives on mid-dimensional ($\mathbb{R}^n = 20, \ldots, 50$) use cases, encompassing both benchmark functions and real-world problems. The objectives of this work are the following:

1. Investigate changes in sampling, preselection, and weighting procedures in the setting of the Standard EDA;

2. Implement a variant of surrogate-assisted CMA-ES;

3. Apply modifications to the surrogate-assisted CMA-ES to increase its sample efficiency;
4. Optimize the hyperparameters of a humanoid robot gait simulation with the goal of minimizing energy consumption in a real-world setting.

3 Estimation of distribution algorithms

All algorithms were implemented in MATLAB and use helper functions from the Statistics and Machine Learning Toolbox. All Gaussian processes (GP) were implemented using the Gaussian processes for machine learning (GPML) toolbox [30]. The CMA-ES implementations take as their scaffold the implementation from the CMA-ES in MATLAB toolbox [31].

3.1 Standard estimation of distribution algorithm

The Estimation of distribution algorithm (EDA) family belongs to the class of stochastic heuristic search algorithms. They store a set of solutions, denoted population when spoken of collectively and individuals when spoken of singularly. Due to their stochastic nature, EDAs are less susceptible to being left stuck at local optima than deterministic search algorithms [15]. They work by sampling new individuals $\mathbf{x}_k$ from a generative model describing the probability distribution $\mathcal{D}$ parametrized by some θ, which is a function of the current population [23].

$$\mathbf{x}_k \sim \mathcal{D}(\theta), \theta = f(\{\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}\}) \quad (1)$$

This estimation procedure, in contrast to finite differences, amortizes the effects of noise in function evaluations by increasing the distance η between them. The error between the noisy gradients of $f(x)$ and the noiseless $h(x)$ has a finite difference component $L\eta^2$ and a noise component $\frac{\epsilon(x;\eta)}{\eta}$ [15].

$$||\nabla f(x) - \nabla h(x)||_\infty \leq L\eta^2 + \frac{\epsilon(x;\eta)}{\eta} \quad (2)$$

Increasing η is most beneficial when evaluations are results of, for example, stochastic simulations. These will always contain a random error component, which is expected to dominate [15]. In expectation, (2) aligns with the direction of the natural gradient and can be used to approximate curvature information [32]. Once a group of individuals is sampled, it is evaluated against the cost function. Then failures are pruned, and surviving individuals are recombined with the existing ones to form a new population. Algorithm 1 formalizes the standard EDA.

Algorithm 1: The standard EDA

Input: n

```
// Initialize
```
$\mathbf{C} \leftarrow \mathbf{I}_n$;
$\mathbf{m} \leftarrow 1_n$;
$\lambda \leftarrow f(n)$;
$\mu \leftarrow f(\lambda)$;

```
// Algorithm loop
```
while $g \leq g_{\max}$ **do**
  $\mathbf{L} \leftarrow \mathrm{Cholesky}(\mathbf{C})$;
  $\mathbf{z}_{k..\lambda} \sim \mathcal{N}(0, \mathbf{I}_n)$;
  $\mathbf{x}_{k..\lambda} \leftarrow \mathbf{L}\mathbf{z}_k + \mathbf{m}$;
  `// CalcParents`
  $\mathbf{y}_{k..\mu} \leftarrow \mathrm{CostFunction}(\mathbf{x}_{k..\lambda})$;
  `/* Sort fittest` $\mathbf{x}_k$`, truncate to` μ `points */`
  `// CalcMean`
  $\mathbf{m} \leftarrow \frac{1}{\mu}\sum_{k=1}^{\mu} \mathbf{x}_{k..\mu}$;
  `// AdaptCovMatrix`
  $\mathbf{C} \leftarrow \frac{1}{\mu}\sum_{k=1}^{\mu}(\mathbf{x}_{k..\mu} - \mathbf{m})(\mathbf{x}_{k..\mu} - \mathbf{m})^\top$;

n is the number of dimensions, $\mathbf{C}$ is the covariance of the sampling distribution, λ is the size of the generation, and μ is the size of the parent vector, that is, the points which go on to drive updates to the sampling distribution. $\mathbf{L}$ is the Cholesky decomposition of the covariance,g is the generation index, k is the rolling index, $\mathbf{z}$ the whitened samples, $\mathbf{x}$ the transformed samples, $\mathbf{m}$ the mean vector and $\mathbf{y}$ the cost. Here, the sampling distribution is given by the empirical maximum likelihood Gaussian distribution of the current population. One notable flaw of this ap-

proach is illustrated in Figure 1, which depicts an EDA solving a two-dimensional problem at generation g on the left frame and $g+1$ on the right frame. The dots represent the population. Diagonal lines are isolines of the cost function, whose value decreases linearly in the direction of the upper right corner. The triangle denotes the sampling distribution's empirical mean, and the ellipses the empirical covariance.

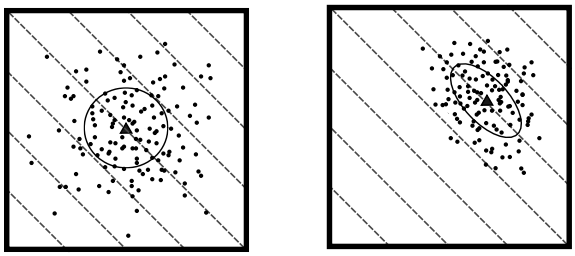

Figure 1. Degeneration of covariance matrix after one generation. Adapted from [32].

As the mean in the maximum likelihood case is the minimizer of the variance in the sampled population, in expectation, the sampling distribution constantly shrinks in the way shown in Figure 1. It shows a sampling distribution whose principal axes are orthogonal to the direction of the gradient of the cost function, incentivizing exploration in a direction with little promise of improvement. This is a major flaw of the standard EDA, making its application inadvisable in cases where the initial sampling distribution does not encompass the optimum. The algorithm keeps running until a certain stopping condition is met. This could be a consecutive failure to improve the cost of the best solution, a maximum prescribed number of iterations having passed, or the covariance matrix having become close to degenerate. Some high-level modifications to this type of algorithm include:

1. Carrying over parameter information from previous generations;
2. Changing the generative model to other distributions;
3. Changing the way the population is replaced between generations. Examples of this are elite strategies [25], where individuals who have held a high fitness value for several generations are kept in the population for a set number of iterations instead of being resampled;
4. Purposefully shrinking the sampling distribution of the generative model if many successive generations fail to improve.

3.2 Mahalanobis, resampling, fitness-based EDA

In this work, the Mahalanobis, Resampling, Fitness-based EDA (MRF-EDA) is developed. It aims to fix the covariance degeneration problem, increase sample efficiency, and attempt to make the Standard EDA more effective in higher dimensions. Changes made are in summary:

1. Mahalanobis sampling scheme;
2. Fitness-based weighing of solutions;
3. Preselecting solutions based on probability of improvement proxy;
4. Updating after every sample.

These are explained in the following ways:

1. Candidate solutions are generated initially from a unit Gaussian. They are then modified by the application of a partial reverse whitening transformation of the form

$$\mathbf{x} = \sqrt{\frac{\mathbf{z}^{\mathrm{T}}\mathbf{z}}{\mathbf{z}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{z}}}\mathbf{z} + \mathbf{m} \qquad (3)$$

which modifies the norm of the vector without changing its basis vector. The result of this transform is to make sampling isotropic about the angle but not the magnitude of the sampled vectors connecting the sampled points to the mean. This

method of sampling shall be called Mahalanobis sampling due to the similarity between it and the formula for the Mahalanobis distance. A comparison of samples generated through regular Gaussian sampling versus Mahalanobis sampling can be seen in Figure 2;

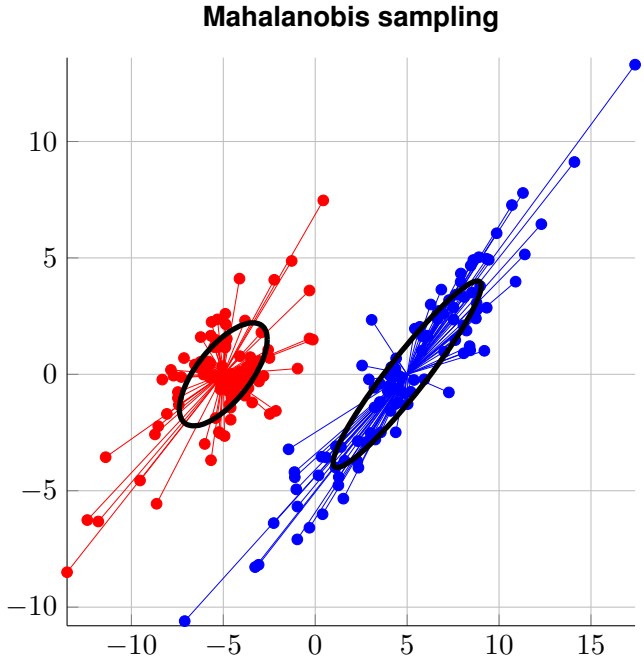

Figure 2. The left cluster was generated with Mahalanobis sampling, and the right cluster with Gaussian sampling; both distributions are highly correlated.

2. A fitness-based weighting scheme defined by the convex combination of all individuals is used instead of taking the maximum likelihood mean and variance of the population to be the sampling distribution. In it, every weight is proportional to the relative cost of it's corresponding individual within the population, with lower costs being assigned larger weights. This is as opposed to other weighing schemes, where all weights are equal or are defined as an exponentially decaying set. This weighing scheme violates affine invariance but is expected to increase convergence speed;

3. A preselection criterion is added. A point is sampled by the fitness-weighted distribution, and the probabilities of it having been generated by the maximum likelihood distribution and the fitness-weighted distribution is calculated. The ratio of the probabilities is calculated, and if it falls under a certain threshold, defined by a hyperparameter, the point is discarded. If a maximum number of attempts fails to surpass the threshold, this criterion is temporarily switched off for generating the next candidate. This criterion is expected to regularly switch off in normal operation if the maximum likelihood and fitness-weighted distribution have a low Kullback-Leibler divergence (KL divergence). This would be the case in areas where the gradient of the cost function is shallow. In areas with steep gradients, it is expected to speed up the search in steep directions. All sampling distributions are kept as Gaussians due to the availability of closed-form solutions for sampling, conditioning, and comparing operations;

4. The number of sampled individuals per generation of the algorithm is reduced to one. The parent population is either unaffected if the cost of the newest individual is higher than that of all members of the current population, or it otherwise replaces the individual with the highest cost and the mean and covariance parameters are updated. In much higher-dimensional problems ($\mathbb{R}^n = 100, \ldots, 500$), this could become computationally expensive, but in the current use cases, all operations, in particular matrix inversions, are computationally cheap in the dimensions encountered.

4 Covariance matrix adaptation - evolution strategy

This work seeks to leverage the covariance matrix adaptation - evolution strategy's (CMA-ES) ability to achieve reliable convergence but mitigate its need for extensive evaluations of the cost function. The standard algorithm and modifications to it are described in the following.

4.1 Standard CMA-ES

The CMA-ES samples the population for a particular generation through

$$\begin{aligned} \mathbf{z}_k^{(g)} &\sim \mathcal{N}(0, \mathbf{I}_n) \\ \mathbf{x}_k^{(g)} &= \mathbf{m}^{(g)} + \sigma^{(g)} \mathbf{L}^{(g)}\, \mathbf{z}_k^{(g)} \\ \mathbf{C}^{(g)} &= \mathbf{L}^{(g)} \mathbf{L}^{(g)\top} \\ k &= 1, \ldots, \lambda \end{aligned} \tag{4}$$

Here, a multivariate unit Gaussian is adapted to the cost function by being transformed by the step size σ, Cholesky decomposition of the covariance matrix $\mathbf{L}$, and mean $\mathbf{m}$. The CMA-ES differs from the EDA in that it divides direction-dependent information from scale information in $\mathbf{C}$ and σ, respectively. These three terms are updated by different metrics. The mean is updated as per

$$\mathbf{m}^{(g+1)} = \sum_{k=1}^{\mu} w_k\, \mathbf{x}_{k:\lambda}^{(g)} \tag{5}$$

where each weight w_k satisfies $w_k > 0$, $\sum_{k=1}^{\mu} w_k = 1$. The solutions $\mathbf{x}_{k:\lambda}$ are ranked in order of increasing cost. Information from previous generations for the adaptation of σ and $\mathbf{C}$ is retained and gradually updated through the evolution paths $\mathbf{p}_\sigma$ (6) and $\mathbf{p}_c$ (7)

$$\begin{aligned} &\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(g)} + \\ &\sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\, \mathbf{C}^{(g)-1/2} \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \end{aligned} \tag{6}$$

$$\begin{aligned} &\mathbf{p}_c^{(g+1)} = (1 - c_c)\mathbf{p}_c^{(g)} + \\ &h_\sigma^{(g+1)} \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\, \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \end{aligned} \tag{7}$$

where c_σ and c_c are learning rates, μ_{eff} is a normalization constant which depends on the chosen weights, and h_σ is the Heaviside indicator, which serves to stall the update if certain conditions are met. The paths represent a cumulation of steps of $\mathbf{m}$ normalized by σ, exponentially decaying the relevance of past steps across generations.

Step-size Control

The step size σ is updated via cumulative step-size adaptation

$$\sigma^{(g+1)} = \sigma^{(g)} \cdot \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{(g+1)}\|}{\mathbb{E}\|\mathcal{N}(0, \mathbf{I}_n)\|} - 1\right)\right) \tag{8}$$

where d_σ is a damping factor. It compares the length of the evolution path to that of the expected length it would have if it were a random walk, increasing the step size if the search is consistently proceeding along the same direction and penalizing it otherwise.

Covariance Matrix Adaptation

The covariance matrix is updated using two update mechanisms. The rank-μ update uses the empirical covariance of the current generation. The rank-one update uses the cumulation from the evolution path (7).

$$\begin{aligned} \mathbf{C}^{(g+1)} &= (1 - c_1 - c_\mu)\mathbf{C}^{(g)} \\ &\quad + c_1\, \mathbf{p}_c^{(g+1)} \mathbf{p}_c^{(g+1)\top} \\ &\quad + c_\mu \sum_{k=1}^{\mu} w_k\, \mathbf{u}_{k:\lambda}^{(g)} \mathbf{u}_{k:\lambda}^{(g)\top} \\ \mathbf{u}_{k:\lambda}^{(g)} &= \frac{\mathbf{x}_{k:\lambda}^{(g)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \end{aligned} \tag{9}$$

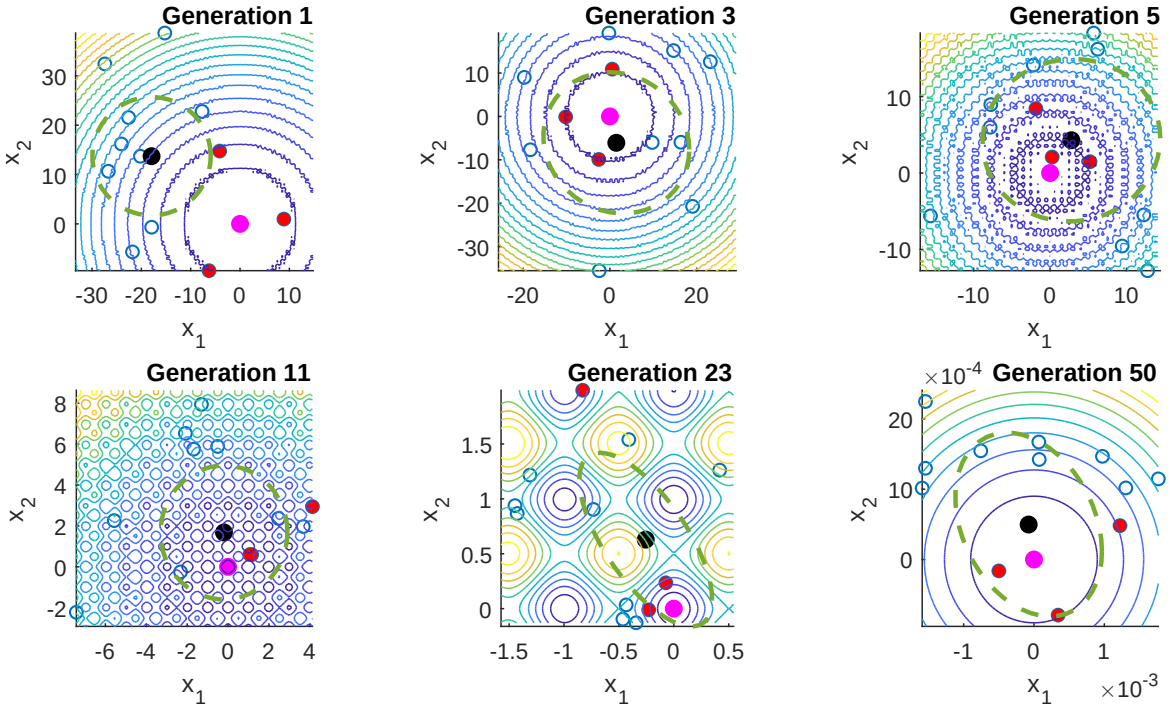

Figure 3. CMA-ES optimizing a Rastrigin function in 2D.

where c_1 is the learning rate for the rank-one update term and c_μ the learning rate for the rank-μ term. The algorithm can be visualized in Figure 3. The large magenta circle represents the global optimum. The large black circle bound by the green ellipse represents the mean and covariance matrix of the sampling distribution. Small circles all represent solutions. Red ones represent the solutions with the lowest cost, chosen to be in the μ vector, whereas hollow blue ones represent solutions that will be discarded. The algorithm can be seen guiding the mean vector close to the global optimum and reducing the size of the sampling distribution as it converges to the global optimum.

4.2 Surrogate-assisted CMA-ES

One significant drawback of CMA-ES is its sample inefficiency, which can be remedied by incorporating a surrogate model. Its role is to assess the relative fitness rankings within a population without having to explicitly evaluate the fitness of every point in the true cost function [27]. This can be done by creating and updating a computationally cheap-to-query surrogate model of the cost function in parallel with the evolution of the CMA-ES. There are many ways of incorporating surrogates and different types of surrogate models, including quadratic models [26], Support vector machines (SVM) [33], and Gaussian processes (GP) [34]. The following is a primer on GP models and sparsity-inducing assumptions, which serve to make them computationally cheaper.

Gaussian process regression

GPs, as seen in (10), are a mathematical object that generalizes the Gaussian probability distribution to the domain of stochastic processes instead of scalar- or vector-valued random vari-

ables [35]. As a stochastic process is continuous for every $\mathbb{R}^{\mathbb{R}}$ number along a particular measure, it can be useful to conceive of a GP as a Gaussian random variable with uncountably infinite dimensionality [35]. Formally, GPs are defined by two functions, the mean $m(\mathbf{x})$ and the covariance function $\mathbf{K}$. The latter of which is positive definite, and consists of an analytic kernel $k(x, x')$ and a noise term $\sigma_n^2\mathrm{I}$

$$\begin{aligned} f(\mathbf{x}) &\sim \mathcal{GP}(m(\mathbf{x}), \mathbf{K}) \\ \mathbf{K} &= k(\mathbf{x}, \mathbf{x}') + \sigma_n^2\mathbf{I} \end{aligned} \tag{10}$$

A particularly prolific kernel is the Matern 5/2 kernel

$$\begin{aligned} k(\mathbf{r}) &= \rho^2 \left(1 + \frac{\sqrt{5}\mathbf{r}}{\ell} + \frac{5\mathbf{r}^2}{3\ell^2}\right) e^{-\sqrt{5}\mathbf{r}/\ell} \\ \mathbf{r} &= ||\mathbf{x} - \mathbf{x}'|| \end{aligned} \tag{11}$$

where ρ is the output variance and ℓ is the length scale. It does not unnecessarily impose an unreasonable smoothness assumption on the sample paths and does not induce the problem of concentration of measure, a phenomenon wherein a random variable that depends on the combination of too many independent variables becomes essentially constant [36]. Remedying this is particularly important when representing high-dimensional phenomena using a GP [36]. GPs can be used to make predictions at unseen test points using

$$\begin{aligned} f_* \mid \mathbf{x}, \mathbf{y}, \mathbf{x}_* &\sim \mathcal{N}(\mathbf{m}_*, \mathbf{K}_*) \\ \mathbf{m}_* &= m(\mathbf{x}_*) + k_*^\top \mathbf{K}^{-1}(\mathbf{y} - m(\mathbf{x})) \\ \mathbf{K}_* &= k(x_*, x_*) - k_*^\top \mathbf{K}^{-1} k_* \\ k_* &= \left[k(x_*, x_1), \ldots, k(x_*, x_j)\right]^\top \end{aligned} \tag{12}$$

where the subscript $*$ denotes an unseen test point and f the latent prediction output before noise is considered. If the prediction targets are known, this can be reversed in order to estimate which kernel hyperparameters θ maximize the marginal likelihood distribution. The canonical form of this maximization, after simplification, is

$$\begin{aligned} \log p(\mathbf{y} \mid \mathbf{x}, \theta) = &-\frac{1}{2}(\mathbf{y} - m(\mathbf{x}))^\top \mathbf{K}_\theta^{-1}(\mathbf{y} - m(\mathbf{x})) \\ &- \frac{1}{2}\log|\mathbf{K}_\theta| \\ \theta^* = &\arg\max_\theta \ \log p(\mathbf{y} \mid \mathbf{x}, \theta) \end{aligned} \tag{13}$$

where the equality holds up to an additive constant. When directly implementing these equations, one sees that there is a severe computational bottleneck in the form of the necessary inversion of the covariance function of the existing data $\mathbf{K}$. The time complexity of such an operation is $\mathcal{O}(j^3)$. This can be reduced to $\mathcal{O}(ju^2)$ by reducing the number of points in the dataset that go on to induce the GP posterior from j to u. Several ways have been proposed to go about this [37],[38],[39], the most theoretically robust of which involves treating the inducing inputs as variational parameters and using them to minimize a divergence between an approximation and the exact GP posterior.

Sparse Variational GP

Variational inference (VI) aims to calculate an approximation of a posterior distribution of some target $\mathbf{y}$ given data $\mathbf{x}$ and parameters θ [40]. $p(\theta \mid \mathbf{y}, \mathbf{x})$, which when expressed using Bayes theorem, takes the form

$$p(\theta \mid \mathbf{y}, \mathbf{x}) = \frac{p(\mathbf{y} \mid \theta, \mathbf{x})\, p(\theta)}{p(\mathbf{y} \mid \mathbf{x})} \tag{14}$$

An approximation is necessary when the marginal likelihood term $p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \theta)p(\theta)\mathrm{d}\theta$ is either intractable or too expensive to be approximated with Monte Carlo integration [41]. This is usually the case for problems where $\mathbf{y}$ has a high dimension. The trade-off consists in the fact that the results obtained by Monte Carlo integration are asymptotically exact, whereas

those from VI are not. VI simplifies (14) by approximating the posterior $p(\mathbf{y} \mid \mathbf{x}, \theta)p(\theta)$ with a user-chosen family of variational distributions $q(\theta)$, examples of which can be Gaussians, Gaussian mixtures, and exponential distributions—families that enable analytical closed-form solutions to the above integral and can easily be sampled from. The exact parametrization of these distributions is then optimized using

$$\log p(\mathbf{y} \mid \mathbf{x}) = \mathcal{L}(q) + \mathrm{KL}(q(\theta) \,\|\, p(\theta \mid \mathbf{y}, \mathbf{x})) \tag{15}$$

which simplifies down to

$$\mathcal{L}(q) = \mathbb{E}_{q(\theta)}[\log p(\mathbf{y} \mid \theta, \mathbf{x})] - \mathrm{KL}(q(\theta) \,\|\, p(\theta)) \tag{16}$$

where the evidence lower bound (ELBO) $\mathcal{L}(q)$ is an equivalent optimization objective up to an additive constant, of attempting to minimize the KL divergence between the proposed variational distribution and the real posterior [41]. This allows one to not compute the intractable $\log p(\mathbf{y}|\mathbf{x})$. In the setting of GPs, the posterior distribution being approximated is the approximate posterior $p(f \mid \mathbf{y}, \mathbf{x}, \mathbf{x}_u, \theta)$ where $\mathbf{x}_u$ are inducing points for the kernel. Arguments of the optimization problem are the kernel hyperparameters and the location of the inducing points. The kernel parameters can be optimized using maximum likelihood, much like in the dense case in (13) if the inducing inputs are fixed a priori. The number of inducing points is a user-chosen hyperparameter, and their optimal locations are given by the following variational method. It minimizes a lower bound to the KL divergence between the variational distribution $q(f)$, created by specifying all kernel hyperparameters and inducing input locations, and the exact posterior defined only on the test function values $p(f|\mathbf{y})$. Once simplified, the optimization takes the form

$$\begin{aligned} \mathcal{L} &= \log \mathcal{N}(\mathbf{y} \mid 0, \mathbf{Q}_{\mathbf{xx}} + \sigma_n^2 \mathbf{I}) \\ &\quad - \frac{1}{2\sigma_n^2} \mathrm{tr}(\mathbf{K}_{\mathbf{xx}} - \mathbf{Q}_{\mathbf{xx}}) \\ \mathbf{Q}_{\mathbf{xx}} &= \mathbf{K}_{\mathbf{x}\mathbf{x}_u} \mathbf{K}_{\mathbf{x}_u\mathbf{x}_u}^{-1} \mathbf{K}_{\mathbf{x}_u\mathbf{x}} \end{aligned} \tag{17}$$

This is best done with block coordinate descent, first optimizing the kernel hyperparameters and then the inducing inputs to avoid overfitting and noisy joint gradients [39]. This all has the effect of eliciting a compressed representation of the dense GP

4.3 Gaussian process-CMA-ES

The Gaussian process-CMA-ES (GP-CMA-ES) modifies the standard CMA-ES by training a GP surrogate of the cost function after every generation, using it to predict the next population's fitness, and preselecting the best solution candidates for evaluation in the real cost function. Algorithm 2 formalizes this notion.

The buff variable denotes a buffer of real cost function evaluations that are not used in the update of the CMA-ES equations. It allows for leniency in the surrogate's ability to classify points, as a wrongly ordered preselection by the surrogate can still be remedied to an extent by the subsequent re-evaluation and re-ordering step. Such a buffer is valuable as more function evaluations allow for a more populated data repository D, leading to more accurate surrogates in subsequent generations. The GP surrogate is parametrized with a Matern 5/2 kernel with automated relevance determination (ARD) as in (11), but with a separate length scale per dimension. A quadratic mean function is used, and costs are assigned to candidate solutions using a lower confidence bound acquisition function. A quadratic mean is chosen as it prevents the large optimistic inaccuracies that a zero or linear mean would induce. If some of the mean's hyperparameters are specified,

Algorithm 2: GP-CMA-ES

Input: $n, w_{k..\mu}, \mathbf{x}$

```
// Initialize
```
$\mathbf{C} \leftarrow \mathbf{I}_n$;
$\lambda \leftarrow (4 + 3\log n)$;
$\mu \leftarrow \lambda/(2 \dots 4)$;
buff $\leftarrow (0.1 \dots 0.4)\mu$;
$\sigma \leftarrow 0.3(\mathbf{x}_{\max} - \mathbf{x}_{\min}))$;
$D^g \leftarrow \emptyset$;
CMAESUpdate$(\mathbf{m}, \mathbf{C}, \sigma)$ (Eqs. (5)–(9))
SurrogatePredict$(\mathcal{GP}, \mathbf{x})$ (Eq. (12))

```
// Algorithm loop
```
while $g \leq g_{\max}$ **do**
  if $D^g \neq \emptyset$ **then**
    $(\mathcal{GP}) \leftarrow$ TrainSurrogate(D^g);
  $\mathbf{L} \leftarrow$ Cholesky$(\mathbf{C})$;
  $\mathbf{z}_{k..\lambda} \sim \mathcal{N}(0, \mathbf{I})$;
  $\mathbf{x}_{k..\lambda} \leftarrow \sigma\mathbf{L}\mathbf{z}_k + \mathbf{m}$;
  `// Preselection`
  $\mathbf{v}_{k..\lambda} \leftarrow$ SurrogatePredict$(\mathcal{GP}, \mathbf{x}_{k..\lambda})$;
  `// Sort increasing cost and truncate to` $\mu +$ `buff points`
  $\mathbf{x}_{k..(\mu+\text{buff})} \leftarrow$ Top$(\mathbf{v}_{k..(\mu+\text{buff})})$;
  `// Evaluate parents`
  $\mathbf{y}_{k..(\mu+)} \leftarrow$ CostFunction$(\mathbf{x}_{k..(\mu+\text{buff})})$;
  `// Sort increasing cost and truncate to` μ `points`
  $(\mathbf{x}_{k..\mu}, \mathbf{y}_{k..\mu}) \leftarrow$ Top$(\mathbf{y}_{k..\mu})$;
  `// update CMA-ES parameters`
  $(\mathbf{m}, C, \sigma) \leftarrow$ CMAESUpdate$(\cdot)$;
  `// Update repository`
  $D^g \leftarrow$
  $D^{g-1} + (\mathbf{x}_{k..(\mu+buff)}, \mathbf{y}_{k..(\mu+buff)})$;

the prior of a GP with a quadratic mean is an increasing function in areas of low data. This disincentivizes undue exploration under high uncertainty. A lower confidence bound acquisition function incorporates variance information and incentivizes targeted exploration in areas where said variance is large. Two further modifications were also explored. Firstly, D can be made to include every single evaluated solution or restrict itself to solutions within a trust region defined from the center of the sampling distribution. If the latter method is elected, the GP surrogate can be trained in whitened space, improving the conditioning of the numerical operations and making the axes of the ARD kernel aligned with the principal axes of the CMA-ES sampling distribution. This increases performance in problems where the coordinates are inseparable. Secondly, in applications with costly function evaluations where some data extracted from previous work is available, the surrogate can be pre-trained with such data. This mitigates the effects of early surrogate unreliability due to low data volume. Expected discrepancies due to, for example, different data collection methodologies can be compensated for by adapting the parameters of the likelihood function of the GP. The local training in whitened space modification was implemented; the pretraining scheme was not.

4.4 GP-CMA-ES with learning rate adaptation

The third algorithm proposed consists of a setup identical to the GP-CMA-ES with one extra accretion, that of a learning rate adaptation (LRA). The rationale motivating this particular LRA consists of exploiting information about the confidence of the GP model in its accuracy in a way that drives the algorithm away from areas that have been sufficiently explored. This is realized by modulating the cumulative step size adaptation from (8) by multiplying it by a term

proportional to the posterior epistemic variance of the GP surrogate. Some previous work has been done on LRA in the area in non-surrogate-assisted versions of CMA-ES [42],[43]. This term is calculated as per Algorithm 3 and multiplied directly to (8).

Algorithm 3: Learning rate adaptation

Input: $\mathbf{L}, \mathbf{m}, \sigma$, number of Monte-Carlo samples M, number of generations in variance history T, current generation g

$\mathbf{z}_{k..M} \sim \mathcal{N}(0, \mathbf{I})$;

$\mathbf{x}_{k..M} \leftarrow \sigma \mathbf{L} \mathbf{z}_{k..M} + \mathbf{m}$;

`// Compute the epistemic variance for each sample`

$\sigma^2_{\mathsf{e}}(\mathbf{x}_{k..M}) \leftarrow \mathcal{GP}(\mathbf{x}_{k..M})$;

$\overline{\sigma^2_{\mathsf{e}}}^{(g)} \leftarrow \frac{1}{M} \sum_{k=1}^{M} \sigma^2_{\mathsf{e}}(\mathbf{x}_k)$;

`// Epistemic variance of the last` T `generations`

$E \leftarrow \{\sigma^{2^{(g..T)}}_{\mathsf{e}} \mid t = g - T + 1, \ldots, g\}$;

$\tilde{\sigma}^2_{\mathsf{e}} \leftarrow \mathrm{median}(E)$;

$R \leftarrow exp(-\frac{\overline{\sigma^2_{\mathsf{e}}}}{\tilde{\sigma}^2_{\mathsf{e}}})$;

$R \leftarrow \mathrm{clip}(R, 0.85, 1.0)$;

return R;

This adaptation scheme uses the predictions of the kernel for a set of Monte Carlo samples drawn from the CMA-ES distribution to estimate the posterior epistemic variance of the samples. This is taken as the model's confidence in its own predictions. Due to the curse of dimensionality, taking the raw epistemic variance prediction would be uninformative, so the adaptation scheme looks at changes in this prediction over generations. This allows the algorithm to decrease its effective confidence in the surrogate at times where it shows high uncertainty by decreasing σ. This prevents the sampling distribution from being misled.

5 Case studies

The algorithms were evaluated firstly on canonical optimization benchmarks [44] and secondly, on a real-world optimization task. It involves the tuning of the hyperparameters of a biologically inspired dead-beat controller (BID) embedded in a whole-body controller (WBC), which generates and stabilizes a running gait of a humanoid robot in a Simulink model [12].

5.1 Canonical black box optimization benchmarks

The functions used to benchmark the performance of black box optimization algorithms test their abilities to perform different feats. The four functions used in this work are represented in Figure 4 and are discussed below. The Rastrigin function 19 consists of the superposition of a parabola and a sinusoid. This function tests the ability of an algorithm to cope with multimodality, i.e. many local minima. The Sphere function 18 tests an algorithm's ability to converge quickly when presented with an uncomplicated objective. The Ackley function 21 retains the multimodality of the Rastrigin and furthermore presents with a shallow basin for most of its domain, except for close to its global minimum, wherein it quickly converges. This tests the aptitude of an algorithm to quickly adapt to drastic changes in the curvature of the function without wasting iterations, in a sense combining the tasks of the Rastrigin and sphere functions within one benchmark. The Rosenbrock function 22 has its optimum at an ill-conditioned region, where the gradient in certain directions is much steeper than in others. This measures an algorithm's ability to effectively adapt to ill-conditioned coordinate systems. An algorithm's performance is measured by a single metric: its ability to decrease the best-so-far cost more quickly, i.e., with fewer true cost function evaluations than other competing algorithms. For these functions, having access to the ground truth and the gradients allows for easy troubleshooting and analysis of drawbacks and errant behavior in the design process of optimization algorithms. Definitions are valid for $\mathbf{x} \to \mathbb{R}^n$

$$f_{\text{Sphere}}(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{x}^2, \tag{18}$$

$$f_{\text{Rastrigin}}(\mathbf{x}) = 10n + \sum_{i=1}^{n} \Big(\mathbf{x}^2 - 10\cos(2\pi\mathbf{x})\Big) \tag{19}$$

$$f_{\text{Ackley}}(\mathbf{x}) = -20\exp\Bigg(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^2}\Bigg) \tag{20}$$

$$-\exp\Bigg(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi\mathbf{x})\Bigg) + 20 + e \tag{21}$$

$$f_{\text{Rosenbrock}}(\mathbf{x}) = \sum_{i=1}^{n-1} \Big(100(x_{i+1} - \mathbf{x}^2)^2 + (\mathbf{x} - 1)^2\Big) \tag{22}$$

Table 1 summarizes qualitative properties of these functions The evaluation of the performance of black box optimizers was done in 30 dimensions [44]. The maximum number of function evaluations par run is 250. No regard is taken with respect to early stopping given stalling. Each function is optimized a total of 10 times with reproducible seeds. The mean vector is initialized as a uniformly distributed random number between -20 and 20. The CMA-ES derivatives are tested with the combination of $\mu = \lambda/4$ and $\mathrm{buff} = 0.4\mu$, and $\mu = \lambda/2$ and $\mathrm{buff} = 0.2\mu$. All other hyperparameters are initialized according to Algorithms 1 and 2. Any hyperparameter not mentioned takes the default value defined in [31].

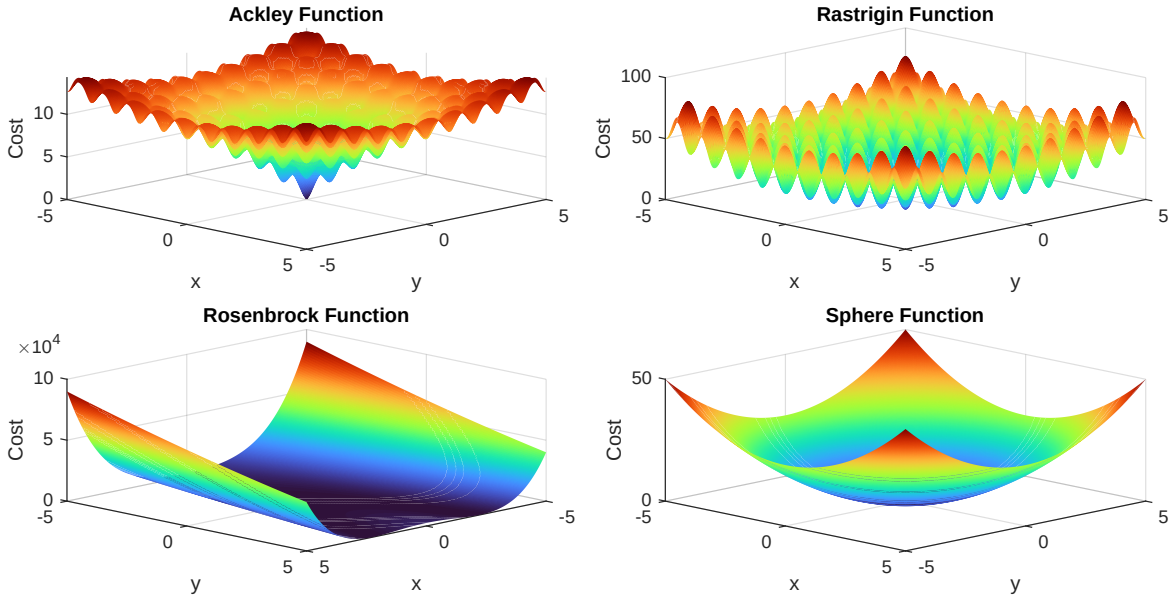

Figure 4. Illustration of benchmark functions.

Table 1. Qualitative properties of the benchmark functions used in this work.

Function	Modality	Separable?	Conditioning / geometry
Sphere	unimodal	yes	well-conditioned, isotropic
Rastrigin	multimodal	yes	highly multimodal, regular local minima
Ackley	multimodal	yes	broad flat region, sharp basin near optimum
Rosenbrock	unimodal	no	ill-conditioned, narrow curved valley

5.2 Gait optimization of a humanoid robot

The second case study consists of optimizing the energy consumption of a full humanoid robot with 28 joint degrees of freedom (DoF) plus 6 free-floating DoF. More specifically, the time-average of the summed absolute joint mechanical power

$$J_{\text{watt}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{J} |\dot{q}_{t,j}(\boldsymbol{\theta})\, \tau_{t,j}(\boldsymbol{\theta})|$$

is defined as the cost function, where N is the number of simulation samples, J the number of actuated joints, $\dot{q}_{t,j}$ the joint velocity, and $\tau_{t,j}$ the commanded joint torque from the high-level controller, all averaged over one episode. The experiment was conducted in a simulation environment. The simulation has the working principle of constraint-consistent integration of forward dynamics of the actuation torque outputs of the WBC [12]. Concretely, this means the low-level controllers (LLC) compute torque commands that do not violate constraints of allowed joint velocities and positions. The BID controls the positions, velocities, and forces on the center of mass (CoM) using spatially independent polynomials when the robot has ground contact. The WBC takes inputs from the BID and uses inverse dynamics to ensure higher-level tasks, such as torso orientation and desired leg trajectories, are fulfilled to an optimal degree by encoding all tasks into a quadratic program. All three controllers have hyperparameters to be optimized. The exact set of 32 parameters

optimized can be seen in the Appendix.

It is expected that the parameter space contains huge swathes of values for which the simulation diverges. The optimization algorithms have no active acknowledgement of these constraints; initializations are kept in regions far away from bounds, as optimal values for the hyperparameters are not expected anywhere near them. Divergent runs are given a maximum cost of 10,000 Watts and are still included in the GP and update equations. The values of all hyperparameters are normalized to between 0 and 1, and all optimization happens in this standardized space. This allows for control of the initializations of the mean vector of the sampling distribution. As a very ill-conditioned and rapidly varying cost function is expected and experimental budget limited, the only hyperparameters tested are $\mu = \lambda/2$ and $\mathrm{buff} = 0.2\mu$, as higher values of λ are known to work better in harder problems [32]. The experiment consisted of the average of three runs per algorithm. The first run took as the starting point, the vector of the previously best-known hand-tuned hyperparameters, the second increased the magnitude of each component of this best vector by 10%, and the third by 20%. The step size was defined as 30% of the measure spanning the feasible hyperparameter space. This ensures that the presumed optimum remained within a 3 standard deviation ellipsoid of the sampling distribution's mean in all cases. The robot is simulated for the equivalent of four seconds, wherein the robot first undergoes an acceleration phase before reaching its maximum speed. The gait has a running pattern and happens in a straight line. The wall clock time of the simulation ranges between 60 and 120 seconds per run.

6 Results and discussion

All algorithms were tested on the four functions detailed in Chapter 5 and the best performers therein were implemented and tested in the simulation. Candidates were the standard EDA, MRF-EDA, CMA-ES, and then four variants of GP-CMA-ES. Those consist of the four possible combinations of dense versus sparse surrogate GP model and the presence versus absence of the LRA term. The combination of dense surrogate plus LRA and sparse surrogate with no LRA did not perform as well as the other two combinations irrespective of the cost function being optimized. Their performance was worse than that of the CMA-ES while using more function evaluation resources and wall clock computational time. This is thought to be due to the sparse surrogate's regularization of the posterior epistemic when compared to the dense surrogate. this makes changes in the variance across generations less noisy when estimated using Monte Carlo, better informing the LRA term. When the LRA is not present, the same regularization is thought to make the surrogate underfit the cost function, leading to misinformed rankings. Due to the fact that the LRA constrains the step size when uncertainty is high, it is hypothesized that eventual misrankings have less of an effect than they would were the step size not constrained. The most promising algorithms, the CMA-ES, the GP-CMA-ES with dense surrogate, and LRA-GP-CMA-ES with sparse surrogate, were further implemented in the simulation. Results are comparative, and the CMA-ES is used as the benchmark. The evaluation criterion is the best so-far-cost versus the number of true evaluations of the cost function.

6.1 MRF-EDA

Mahalanobis sampling solved the issue of covariance degeneration as illustrated by Figure 1. The principal axes of the MRF-EDA remain aligned with the search direction, whereas those of the standard EDA collapse to small values in the direction of optimal search, as seen in Figure 5.

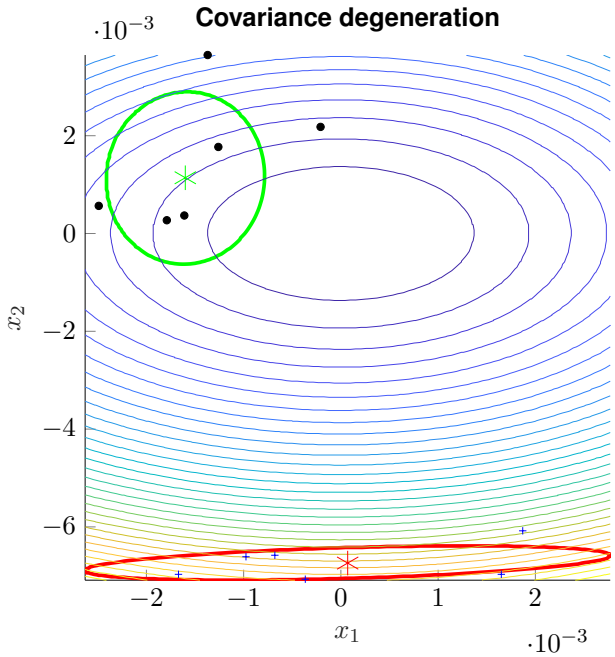

Figure 5. Sampling distributions and populations of standard EDA and MRF-EDA on Sphere function.

The green and red ellipses represent a 2 standard deviation distance from the mean of the sampling distributions of the MRF-EDA and Standard EDA, respectively. Stars represent the mean, blue crosses represent the population of the standard EDA and black dots the population of the MRF-EDA. Figure 5 is a snapshot of a late-stage optimization run where this result was observed.

When comparing the fitness-based weighting scheme with the geometric, fixed-weight scheme, the former slightly outperformed the latter on

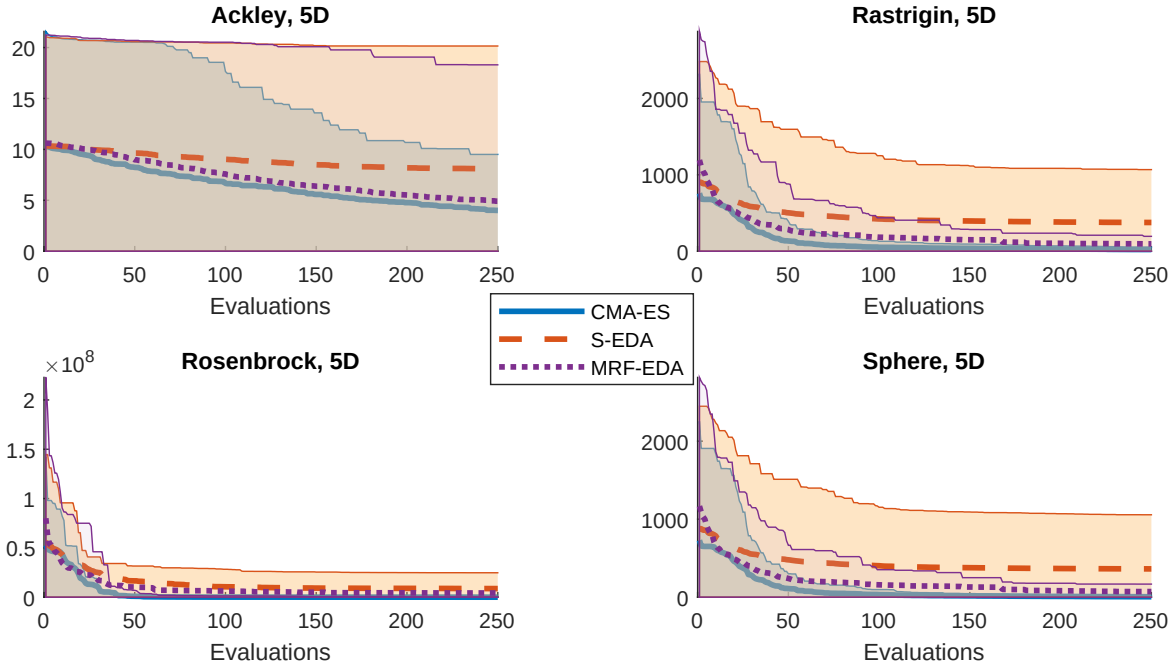

Figure 6. Mean best-so-far cost, 10th and 90th quantile band across 10 runs of both EDAs and CMA-ES with $\mu = \lambda/2$.

Sphere and Rastrigin and was outperformed on Ackley and Rosenbrock. This is due to the conditioning of the function around the solution being good on the functions it performed well in and bad in the ones it did not. This explanation is confirmed by observing some performance degradation, visible as a wider quantile interval in the first 20 evaluations in comparison to that of the standard EDA on an ill-conditioned Rastrigin function. Its ratio between smallest and largest cooordinate is 10^6. The plots for this experiment can be found in the Appendix.

These improved results due to fitness based weighting are expected in smooth functions like the Sphere, due to the fact that in unimodal, convex functions, elite strategies are not as easily mislead. It was also observed that one point with a disproportionate improvement in fitness over the previous pool due to the random nature of the sampling, can severely shift the search distribution to a better overall search area. This would explain the good performance on the well-conditioned Rastrigin function, as the algorithm is able to escape local minima. Updating the search distribution with direct fitness information, even though it leads to the violation of affine invariance [24], can be considered a viable modification in select, non-ill-conditioned cases.

The performance of the EDA family was lackluster in comparison to the CMA-ES on higher-dimensional problems. This can be seen in an experiment in a 5-dimensional Rastrigin function, illustrated in Figure 6. Its mean best-so-far cost lies outside the 10th and 90th quantile band of CMA-ES at 250 evaluations in all benchmark functions except for Ackley. It does however perform much better than the stadndard EDA accross the board, achieving a best cost four times lower at the limit of 250 evaluations on sphere, the best case comparison in favor of the MRF-EDA between the two algorithms. The MRF-EDA was not a serious contender for global optimization in the simulation use case without a favorable initialization. It performed acceptably when implemented in the simulation,

so long as the size of its sampling distribution was kept to 10% of the range of each hyperparameter and its mean vector was initialized to the values of the best known hyperparameters found through hand tuning. costs close to the optimum 520 Watts were returned at the end of 250 evaluations. The exploration of Mahalanobis sampling and fitness-based weights in CMA-ES is left for further work. The resampling/preselection scheme in the MRF-EDA is not suitable to be implemented in CMA-ES, as it would strongly violate the uniformity of sampling assumption, leading to divergence.

6.2 GP-CMA-ES and LRA-GP-CMA-ES

The GP-CMA-ES, both with and without the LRA, achieves the goal of being faster than the CMA-ES accross all values of true cost function evaluations on the benchmarking functions. This can be seen in Figures 7 and 8. Behavior across the benchmark varies strongly depending on the population hyperparameters μ and buff. in the case where $\mu = \lambda/2, \mathrm{buff} = 0.2\mu$, the LRA-GP-CMA-ES achieves a lower total cost in the limit of 250 evaluations in Sphere and Rastrigin. All algorithms tie on this metric on Rosenbrock, and both CMA-ES and LRA-GP-CMA-ES performe woese than the GP-CMA-ES on Ackley, with the means of both other algorithms falling outside the GP-CMA-ES' 90th quantile. the LRA-GP-CMA-ES usually begins the run with an advantage in early iterations over all other algorithms for about 50 evaluations. The number of inducing inputs of the Sparse GP did not have any effect on convergence $u \in \{50, 100, 500\}$. Different trends emerge when Using $\mu = \lambda/4, \mathrm{buff} = 0.4\mu$. These settings increase the performance gap between the GP-CMA-ES and the CMA-ES in favor of the former from, for example, 26% when using $\mu = \lambda/2$ to 38% when using $\mu = \lambda/4$ on the Rastrigin function at a count of 100 evaluations. This result occurs due to the number of evaluations saved per generation. with $\mu = \lambda/4$, 6 true cost function evaluations take place versus 8 when $\mu = \lambda/2$. The LRA-GP-CMA-ES suffers performance degradation on Sphere and Rastrigin with $\mu = \lambda/4$, and fails to continue improving in the limit of 250 evaluations with these settings. This is evident by the wide quantile interval at 250 function evaluations, which is not present when $\mu = \lambda/2$. These settings do however enable it to perform much better on Ackley and Rosenbrock, with it being the best performer in the latter function accross all algorithms tested when $\mu = \lambda/4$. The concrete takeaway for the hyperparameter settings of the LRA-GP-CMA-ES is to set $\mu = \lambda/4$ when functions have a global quadratic structure (Sphere, Rosenbrock) and to set $\mu = \lambda/2$ to allow for a more accurate surrogate to develop when optimizing ill conditioned problems (Ackley, Rosenbrock).

These results seem to indicate that surrogate quality is adequate to guide search through its preselection mechanism, evident by both surrogate assisted algorithms improving upon the CMA-ES's performance in the benchmark functions. The LRA seems to achieve it's effect of slowing down the search at times when uncertainty is high. In convex benchmarks like Sphere, the kernel quickly learns and extrapolates the smooth cost function's geometry, allowing for the LRA term to stay high even in regions with low data due to the low epistemic uncertainty of the extrapolation. In other cases, such as in the Rastrigin function, there are more fluctuations in the LRA term, which could be explained by the inability of the kernel to learn the global structure. This inability is due to multimodality creating high epistemic uncertainty in regions with very little data, as the mean and kernel's prior have no mechanism to account for it. As mentioned before, the LRA-GP-CMA-ES has an advantage in most cost functions in the first 30-50 iterations. Further work could attempt to pinpoint heuristics to de-

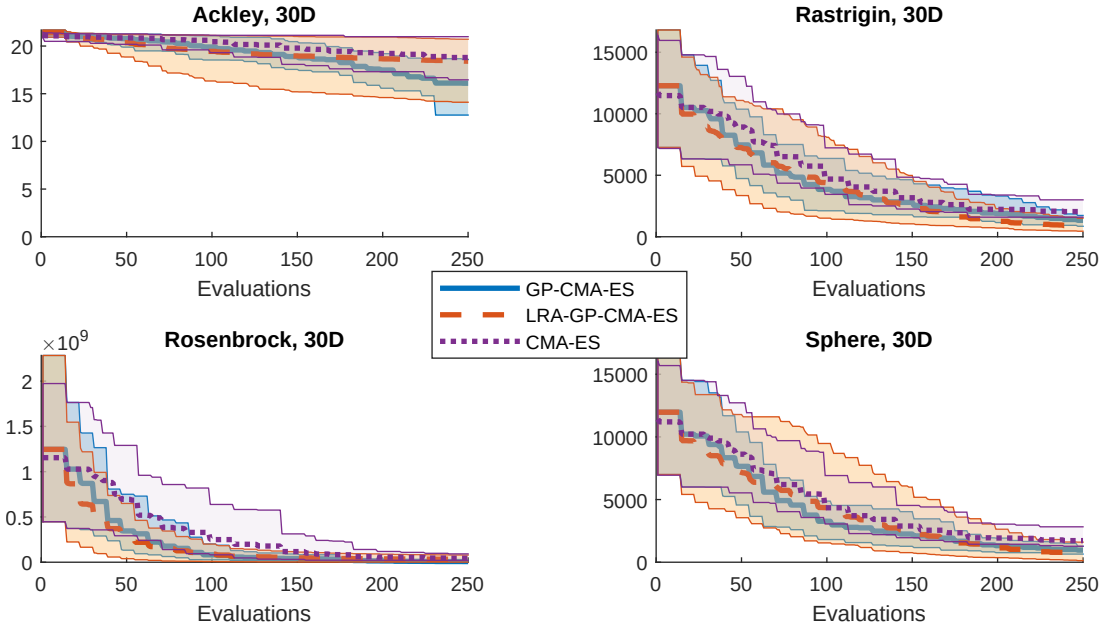

Figure 7. Mean best-so-far cost, 10th and 90th quantile band across 10 runs per algorithm. True cost function evaluations on static benchmarks, $\mu = \lambda/2, \mathrm{buff} = 0.2\mu$.

termine the point in the optimization run when the term stops being useful as a function of the cost function's properties. This would allow for the creation of a further version of the LRA-GP-CMA-ES algorithm, which starts with the learning rate on and disables it after a set number of iterations, or conducts some more elaborate scheduling. It is worth noting that it is not possible to directly calculate the exact theoretical step size the algorithm would have had at a particular generation if the LRA were disabled. This is explained by the recursive nature of the evolution paths in (6) and (7). Different populations would have been generated without the LRA, and their exact makeup cannot be analytically determined a priori. The results also anticipate a particular critique, which states that the step size is always just being clamped down to its minimum, as there is never enough information to meaningfully reduce the epistemic variance in a 30-dimensional space. A theoretical counter to this is proposed in section 4.4, and the empirical results speak to the precise opposite. A clamped-down step size would not outperform the algorithm without the LRA in the early stages, as the magnitude of progress per iteration is directly correlated to the magnitude of the step size on smooth, unimodal functions like the Sphere.

6.3 Results in simulation

The results of the simulation demonstrate the ability of the algorithms to determine good hyperparameters with performance comparable to that of hand tuning. The precise initialization of the hyperparameter vector which served as the mean of the sampling distribution had an effect on the final cost, meaning runs initialized with the same mean vector had similar final costs after 250 evaluations. The CMA-ES and GP-CMA-ES struggled due to the ill-conditioned nature of the cost function and extreme levels of cross-over effects between the hyperparameters, something that kept their step sizes high even when starting with the optimum as the

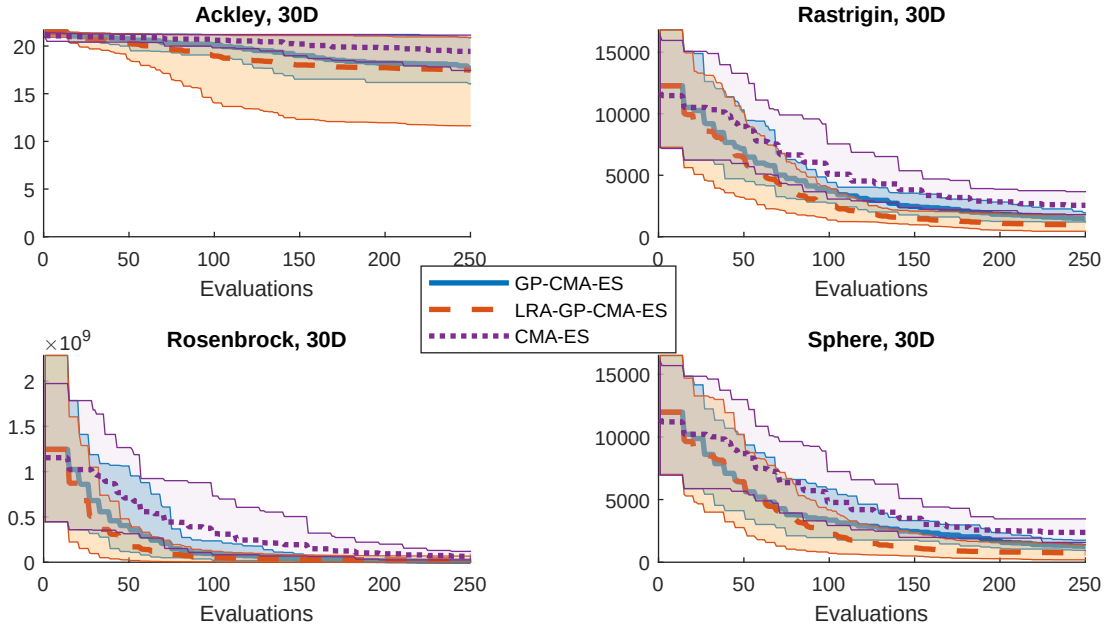

Figure 8. Mean best-so-far cost, 10th and 90th quantile band across 10 runs per algorithm. True cost function evaluations on static benchmarks, $\mu = \lambda/4, \mathrm{buff} = 0.4\mu$.

mean of their sampling distributions. The regularization of the sparse surrogate of the LRA-GP-CMA-ES together with its more aggressive shrinkage of the step size allowed it to model the cost function better than the other two algorithms, whose more fine grained modeling lead to a detriment in their performancet, particularly in the case of the GP-CMA-ES. Step sizes were initialized to 30% of the search width and would either stay constant or grow to 40%. They do, however, always find a usable set of parameters when initialized in the way proposed in Section 5. The CMA-ES and GP-CMA-ES at times generate infeasible hyperparameters throughout the entirity of the optimization run. The LRA-GP-CMA-ES however, stops proposing infeasible solutions after 10 iterations due to the reduction in step size. This reduction in step size leads to a tighter population spread and faster convergence to better hyperparameters, as seen in Figure 9. The LRA-GP-CMA-ES finds a hyperparameter set with the best cost of 474 Watts per episode when previous best cost achieved by hand tuning lay at 520 Watts per episode. Its average results are better than those of the CMA-ES by 10% and those of the GP-CMA-ES by 24%. Knowing that a better set of hyperparameters was found using black box optimization and that the initialization of the algorithms has a reproducible effect on the best achieved cost allows for new experiments to be designed. The definition range of all parameters now can be partitioned, and several starting mean vectors can be chosen with grid search or latin hypercube sampling. This allows for a potentially tractable exhaustive treatment of the parameter space, making sure no promising parameter combinations are overlooked.

The algorithms developed in this work can be implemented in real time tuning with real hardware, as the optimization procedure does not change when going from simulation to reality. Current limitations of pertain to classic issues when transitioning from the simulation of a robot to an experiment on real robot hardware. Certain changes would need to be made to the experiment's scaffolding. Transient effects would

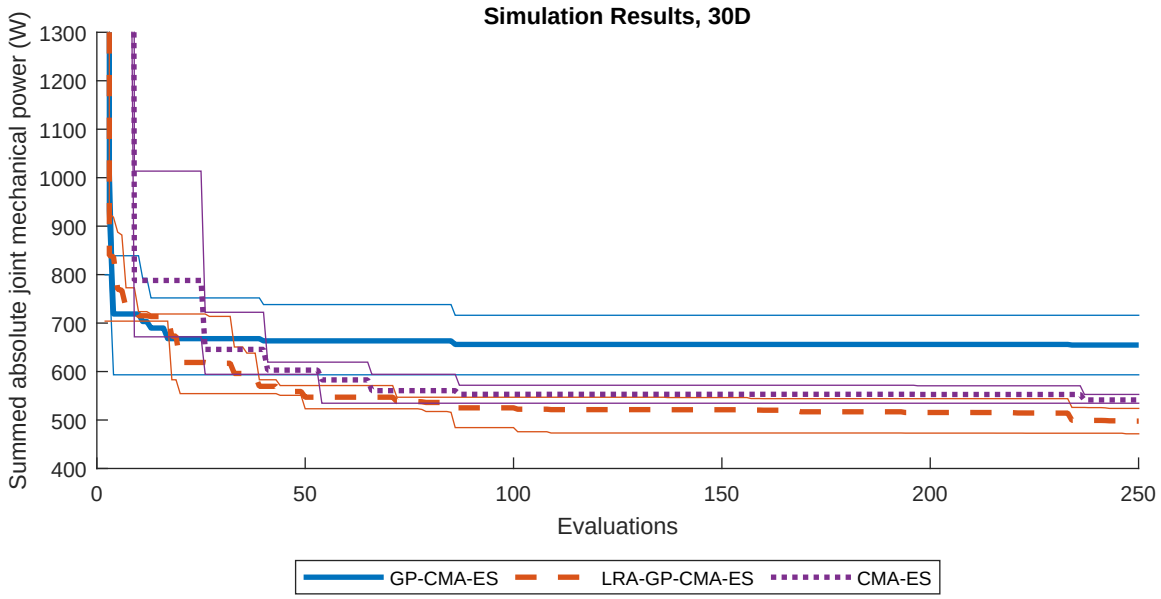

Figure 9. Mean best-so-far cost, 10th and 90th quantile band of 3 runs per algorithm. True cost function evaluations on simulation, $\mu = \lambda/2, \mathrm{buff} = 0.2\mu$.

need to decay between changes in the tested hyperparameters. lastly, new strategies would need to be developed to deal with infeasibility in order to assure safe operation.

7 Conclusion

In conclusion, the modifications to the standard EDA and the CMA-ES both show promise in remedying the shortcomings of those algorithms that they sought to fix. The MRF-EDA solved the problem of covariance degeneration and proved effective on low-dimensional synthetic benchmarks as well as in simulation when initialized with favorable hyperparameters. Both surrogate-assisted CMA-ES versions managed to provide accurate ranking information of cost function evaluations, evident by their faster convergence with a smaller number of true cost function evaluations than the CMA-ES. Wall clock training time for the GP surrogates was negligible when compared to the time taken to conduct simulation experiments. The surrogates showed adequate ability to guide the search even in regions with low data. The LRA mechanism also proved to be an effective development to increase sample efficiency in the 30-dimensional spaces explored. Further work could see the step size being adapted in different ways than the one seen in the present work, with changes encompassing the nature of the modulation and its scheduling across runs. The deployment of the surrogate-assisted algorithms in the simulation yielded positive results, returning hyperparameters comparable to the best set found with hand tuning and, at times, returning sets of hyperparameters with a performance 10% better than that achieved by what were previously thought to be the best hyperparameters for the system. An extension into real-time tuning would need little change to the optimization setup of this work. Instead of running simulation instances episodically, one would use Simulink as the experiment orchestrator, proposing new parameter sets for evaluation, waiting for transient behavior to decay, and conducting measurements of the cost function once steady state is reached. As there are no real-world consequences for infeasible results from the simulation, future work would need to establish a guideline to ensure safe operation of real robot hardware in cases where the chosen hyperparameters return an infeasible task.

Acknowledgments

I would like to thank Dr.Ing Johannes Englsberger for the collaboration and input when developing the algorithms and Dr.Ing Sebastian Stemmler for the proofreading and aid with the manuscript preparation. Additionally, I would like to thank the German Aerospace Center (DLR) for enabling this collaboration. MATLAB versions used: 2024a and 2018b. A free subscription plan to ChatGPT was used to generate simple parts of the MATLAB and LaTeX code used in this thesis.

Bibliography

[1] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (Oct. 2017), pp. 354–359. ISSN: 1476-4687. DOI: 10.1038/nature24270. URL: https://www.nature.com/articles/nature24270 (visited on 05/15/2026).

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2012), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/10.1145/3065386 (visited on 05/15/2026).

[3] A. Auger and N. Hansen. “A Restart CMA Evolution Strategy With Increasing Population Size”. In: *2005 IEEE Congress on Evolutionary Computation*. 2005 IEEE Congress on Evolutionary Computation. Vol. 2. Edinburgh, Scotland, UK: IEEE, 2005, pp. 1769–1776. ISBN: 978-0-7803-9363-9. DOI: 10.1109/CEC.2005.1554902. URL: http://ieeexplore.ieee.org/document/1554902/ (visited on 05/15/2026).

[4] Alejandro Morales-Hernández, Inneke Van Nieuwenhuyse, and Sebastian Rojas Gonzalez. “A survey on multi-objective hyperparameter optimization algorithms for machine learning”. In: *Artificial Intelligence Review* 56.8 (Aug. 1, 2023), pp. 8043–8093. ISSN: 1573-7462. DOI: 10.1007/s10462-022-10359-2. URL: https://doi.org/10.1007/s10462-022-10359-2 (visited on 03/09/2026).

[5] Daniel Lizotte et al. *Automatic Gait Optimization with Gaussian Process Regression.* Pages: 944-949. Jan. 1, 2007. 944 pp.

[6] Han Wang et al. *No More Pesky Hyperparameters: Offline Hyperparameter Tuning for RL*. May 18, 2022. DOI: 10.48550/arXiv.2205.08716. arXiv: 2205.08716[cs]. URL: http://arxiv.org/abs/2205.08716 (visited on 03/09/2026).

[7] Mohaimenul Azam Khan Raiaan et al. “A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks”. In: *Decision Analytics Journal* 11 (June 1, 2024), p. 100470. ISSN: 2772-6622. DOI: 10.1016/j.dajour.2024.100470. URL: https://www.sciencedirect.com/science/article/pii/S2772662224000742 (visited on 05/12/2026).

[8] G. Onori et al. “Adaptive Optimization of Hyper-Parameters for Robotic Manipulation through Evolutionary Reinforcement Learning”. In: (2024). Accepted: 2024-12-04T09:01:27Z. DOI: 10.1007/s10846-024-02138-8. URL: https://re.public.polimi.it/handle/11311/1278425 (visited on 05/15/2026).

[9] Roberto Calandra et al. “An experimental comparison of Bayesian optimization for bipedal locomotion”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014 IEEE International Conference on Robotics and Automation (ICRA). Hong Kong, China: IEEE, May 2014, pp. 1951–1958. ISBN: 978-1-4799-3685-4. DOI: 10.1109/ICRA.2014.6907117. URL: http://ieeexplore.ieee.org/document/6907117/ (visited on 03/09/2026).

[10] Hiroshi Kaminaga, Johannes Englsberger, and Christian Ott. “Kinematic optimization and online adaptation of swing foot trajectory for biped locomotion”. In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012). ISSN: 2164-0580. Nov. 2012, pp. 593–599. DOI: 10.1109/HUMANOIDS.2012.6651580. URL: https://ieeexplore.ieee.org/abstract/document/6651580 (visited on 05/12/2026).

[11] Sait Sovukluk et al. “Realtime Limb Trajectory Optimization for Humanoid Running Through Centroidal Angular Momentum Dynamics”. In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. 2025 IEEE International Conference on Robotics and Automation (ICRA). May 2025, pp. 404–410. DOI: 10.1109/ICRA55743.2025.11127382. URL: https://ieeexplore.ieee.org/abstract/document/11127382 (visited on 05/12/2026).

[12] Johannes Englsberger. “Combining reduced dynamics models and whole-body control for agile humanoid locomotion”. PhD thesis. Technische Universität München, 2016. URL: https://mediatum.ub.tum.de/1311519 (visited on 05/12/2026).

[13] Jia Mian Tan et al. “Hyperparameter optimization: Classics, acceleration, online, multi-objective, and tools”. In: *Mathematical biosciences and engineering: MBE* 21.6 (June 14, 2024), pp. 6289–6335. ISSN: 1551-0018. DOI: 10.3934/mbe.2024275.

[14] James Bergstra and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization”. In: *Journal of Machine Learning Research* 13 (2012).

[15] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization (2nd edition)*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN: 978-0-387-30303-1. DOI: 10.1007/978-0-387-40065-5. URL: http://link.springer.com/10.1007/978-0-387-40065-5 (visited on 03/04/2026).

[16] J. A. Nelder and R. Mead. “A Simplex Method for Function Minimization”. In: *The Computer Journal* 7.4 (Jan. 1, 1965), pp. 308–313. ISSN: 0010-4620, 1460-2067. DOI: 10.1093/comjnl/7.4.308. URL: https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/7.4.308 (visited on 05/09/2026).

[17] Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau. “Simulated Annealing: From Basics to Applications”. In: *Handbook of Metaheuristics*. Ed. by Michel Gendreau and Jean-Yves Potvin. Vol. 272. Series Title: International Series in Operations Research & Management Science. Cham: Springer International Publishing, 2019, pp. 1–35. ISBN: 978-3-319-91085-7 978-3-319-91086-4. DOI: 10.1007/978-3-319-91086-4_1. URL: http://link.springer.com/10.1007/978-3-319-91086-4_1 (visited on 05/15/2026).

[18] Jiapu Zhang. “Simulated annealing: in mathematical global optimization computation, hybrid with local or global search, and practical applications in crystallography and molecular modelling”. In: *arXiv.org* (Aug. 28, 2013). DOI: https://doi.org/10.48550/arXiv.1308.6220. URL: https://arxiv.org/abs/1308.6220v1 (visited on 05/15/2026).

[19] Roman Garnett. *Bayesian Optimization Book*. Bayesian Optimization Book. URL: https://bayesoptbook.com// (visited on 05/12/2026).

[20] Jasper Snoek et al. *Scalable Bayesian Optimization Using Deep Neural Networks*. July 13, 2015. DOI: 10.48550/arXiv.1502.05700. arXiv: 1502.05700[stat.ML]. URL: http://arxiv.org/abs/1502.05700 (visited on 05/15/2026).

[21] Zhitong Xu and Shandian Zhe. *Standard Gaussian Process is All You Need for High-Dimensional Bayesian Optimization*. version: 2. Mar. 7, 2024. DOI: 10.48550/arXiv.2402.02746. arXiv: 2402.02746[cs]. URL: http://arxiv.org/abs/2402.02746 (visited on 03/09/2026).

[22] David Eriksson et al. *Scalable Global Optimization via Local Bayesian Optimization*. arXiv.org. Oct. 3, 2019. URL: https://arxiv.org/abs/1910.01739v4 (visited on 05/12/2026).

[23] Endika Bengoetxea. “Inexact Graph Matching Using Estimation of Distribution Algorithms”. PhD thesis. 2002.

[24] Nikolaus Hansen and Andreas Ostermeier. “Completely Derandomized Self-Adaptation in Evolution Strategies”. In: *Evolutionary Computation* 9.2 (June 2001), pp. 159–195. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/106365601750190398. URL: https://direct.mit.edu/evco/article/9/2/159-195/892 (visited on 05/12/2026).

[25] Dirk V. Arnold and Nikolaus Hansen. “A (1+1)-CMA-ES for constrained optimisation”. In: *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. GECCO ’12: Genetic and Evolutionary Computation Conference. Philadelphia Pennsylvania USA: ACM, July 7, 2012, pp. 297–304. ISBN: 978-1-4503-1177-9. DOI: 10.1145/2330163.2330207. URL: https://dl.acm.org/doi/10.1145/2330163.2330207 (visited on 05/15/2026).

[26] Zyed Bouzarkouna, Anne Auger, and Didier Yu Ding. “Local-meta-model CMA-ES for partially separable functions”. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. GECCO ’11. New York, NY, USA: Association for Computing Machinery, July 12, 2011, pp. 869–876. ISBN: 978-1-4503-0557-0. DOI: 10.1145/2001576.2001695. URL: https://dl.acm.org/doi/10.1145/2001576.2001695 (visited on 05/15/2026).

[27] Zbyněk Pitra et al. “Overview of surrogate-model versions of covariance matrix adaptation evolution strategy”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO ’17. New York, NY, USA: Association for Computing Machinery, July 15, 2017, pp. 1622–1629. ISBN: 978-1-4503-4939-0. DOI: 10.1145/3067695.3082539. URL: https://dl.acm.org/doi/10.1145/3067695.3082539 (visited on 02/26/2026).

[28] Paweł Szynkiewicz. “(PDF) A Comparative Study of PSO and CMA-ES Algorithms on Black-box Optimization Benchmarks”. In: *Research and Academic Computer Networ* (May 4, 2026). DOI: 10.26636/jtit.2018.127418. URL: https://www.researchgate.net/publication/329971785_A_Comparative_Study_of_PSO_and_CMA-ES_Algorithms_on_Black-box_Optimization_Benchmarks (visited on 05/09/2026).

[29] Sander Van Rijn, Carola Doerr, and Thomas Bäck. “Towards an Adaptive CMA-ES Configurator”. In: PPSN 2018. Sept. 8, 2018.

[30] Carl Edward Rasmussen and Hannes Nickisch. “Gaussian Processes for Machine Learning (GPML) Toolbox”. In: ().

[31] Mostapha Heris. *CMA-ES in MATLAB*. Version 2026. URL: https://www.mathworks.com/matlabcentral/fileexchange/52898-cma-es-in-matlab (visited on 05/15/2026).

[32] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*. 2016. DOI: 10.48550/arXiv.1604.00772. arXiv: 1604.00772[cs.LG]. URL: http://arxiv.org/abs/1604.00772 (visited on 05/15/2026).

[33] Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. “Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy”. In: *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. GECCO ’12: Genetic and Evolutionary Computation Conference. Philadelphia Pennsylvania USA: ACM, July 7, 2012, pp. 321–328. ISBN: 978-1-4503-1177-9. DOI: 10.1145/2330163.2330210. URL: https://dl.acm.org/doi/10.1145/2330163.2330210 (visited on 05/15/2026).

[34] Holger Ulmer, Felix Streichert, and Andreas Zell. “Optimization by Gaussian Processes assisted Evolution Strategies”. Series Title: Operations Research Proceedings. Berlin, Heidelberg, 2004. DOI: 10.1007/978-3-642-17022-5_56. URL: http://link.springer.com/10.1007/978-3-642-17022-5_56 (visited on 05/15/2026).

[35] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. 3. print. Adaptive computation and machine learning. Cambridge, Mass.: MIT Press, 2008. 248 pp. ISBN: 978-0-262-18253-9.

[36] Ashish Anil Pawar and Ujwal Warbhe. *Optimizing Bayesian acquisition functions in Gaussian Processes*. Nov. 9, 2021. DOI: 10.48550/arXiv.2111.04930. arXiv: 2111.04930[cs]. URL: http://arxiv.org/abs/2111.04930 (visited on 03/06/2026).

[37] Edward Snelson and Zoubin Ghahramani. “Sparse Gaussian Processes using Pseudo-inputs”. In: *Advances in Neural Information Processing Systems*. Vol. 18. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper/2005/hash/4491777b1aa8b5b32c2e8666dbe1a49Abstract.html (visited on 05/16/2026).

[38] Andrew Naish-Guzman and Sean Holden. “The generalized FITC approximation”. In: *Proceedings of the 21st International Conference on Neural Information Processing Systems*. NIPS’07. Red Hook, NY, USA: Curran Associates Inc., Dec. 3, 2007, pp. 1057–1064. ISBN: 978-1-60560-352-0. (Visited on 05/16/2026).

[39] Michalis Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics. PMLR, Apr. 15, 2009, pp. 567–574. URL: https://proceedings.mlr.press/v5/titsias09a.html (visited on 03/04/2026).

[40] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. “Variational Inference: A Review for Statisticians”. In: *Journal of the American Statistical Association* 112.518 (Apr. 3, 2017), pp. 859–877. ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.2017.1285773. arXiv: 1601.00670[stat]. URL: http://arxiv.org/abs/1601.00670 (visited on 03/04/2026).

[41] Felix Leibfried et al. “A Tutorial on Sparse Gaussian Processes and Variational Inference”. In: *arXiv.org* (Dec. 27, 2020). URL: https://arxiv.org/abs/2012.13962v14 (visited on 05/15/2026).

[42] Gresa Shala et al. “Learning Step-Size Adaptation in CMA-ES”. In: *Parallel Problem Solving from Nature – PPSN XVI*. Ed. by Thomas Bäck et al. Vol. 12269. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 691–706. ISBN: 978-3-030-58111-4 978-3-030-58112-1. DOI: 10.1007/978-3-030-58112-1_48. URL: http://link.springer.com/10.1007/978-3-030-58112-1_48 (visited on 03/02/2026).

[43] Masahiro Nomura, Youhei Akimoto, and Isao Ono. “CMA-ES with Learning Rate Adaptation”. In: *ACM Transactions on Evolutionary Learning and Optimization* 5.1 (Mar. 31, 2025), pp. 1–28. ISSN: 2688-3007. DOI: 10.1145/3698203. arXiv: 2401.15876[cs.NE]. URL: http://arxiv.org/abs/2401.15876 (visited on 05/16/2026).

[44] Ouassim Elhara et al. *COCO: The Large Scale Black-Box Optimization Benchmarking (bbob-largescale) Test Suite*. Mar. 28, 2019. DOI: 10.48550/arXiv.1903.06396. arXiv: 1903.06396[math]. URL: http://arxiv.org/abs/1903.06396 (visited on 03/02/2026).

Appendix

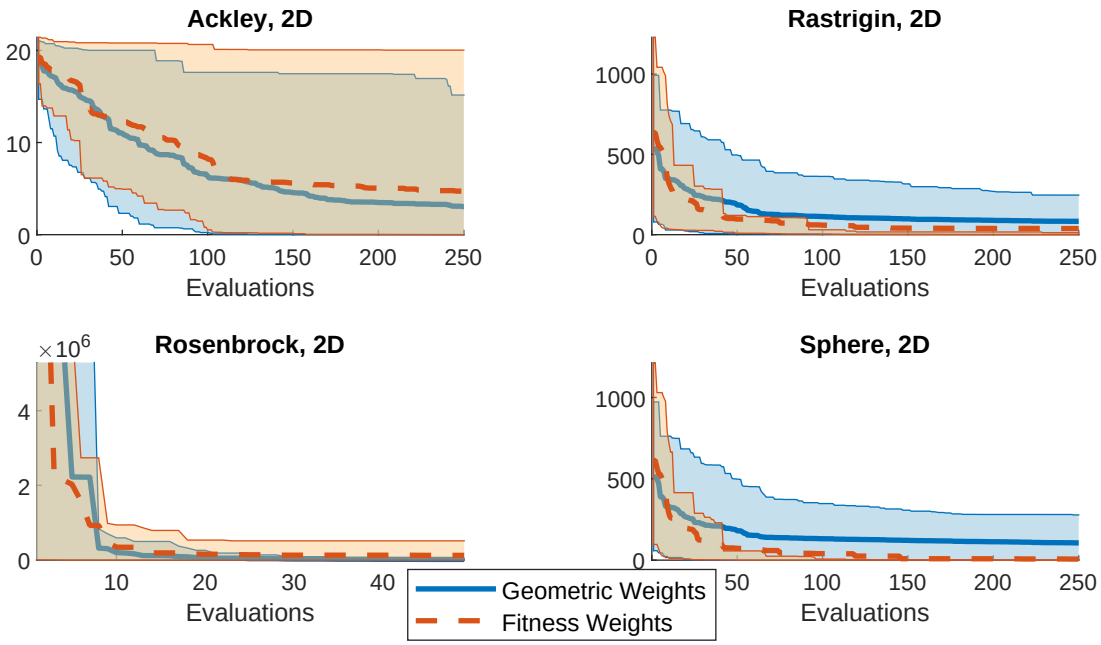

Figure 1: Eof MRF-EDA with precalculated exponentially decaying weights versus fitness-based weights on static benchmarks. Rosenbrock plot is cropped for visibility.

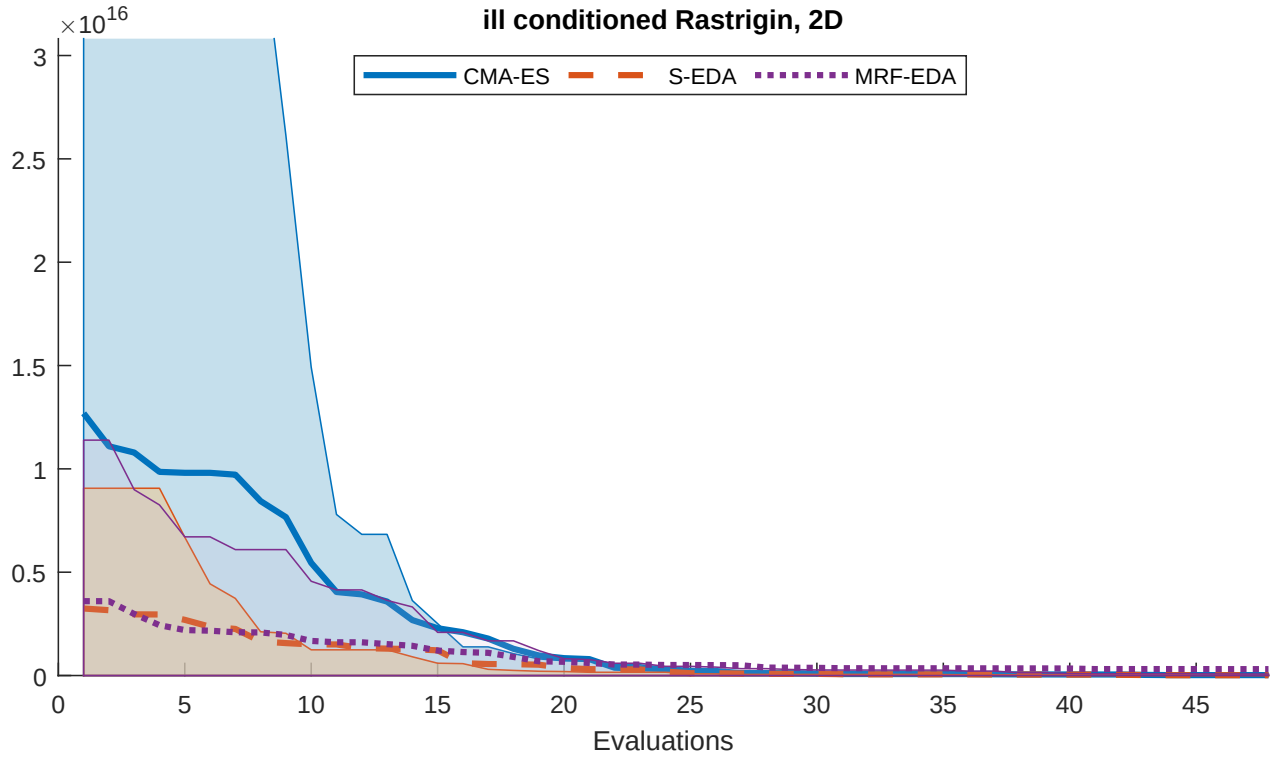

Figure 2: Performance degradation of fitness- based weights on ill conditioned Rastrigin function

Table 1: Humanoid Robot Optimization Hyperparameters and Their Descriptions

Parameter	**Brief Explanation**
desired cadence	Desired stepping frequency or walking rhythm of the humanoid robot.
stance percentage	Percentage of the gait cycle during which the foot remains in ground contact.
Δz TD desired	Desired vertical foot displacement during touchdown to control landing smoothness.
lateral sway offset	Controls side-to-side CoM sway for balance during walking.
desired foot swing height	Desired maximum height of the swinging foot during gait motion.
xy slope	Desired terrain inclination in the horizontal directions.
z slope	Vertical terrain slope or elevation gradient compensation parameter.
desired torso pitch angle	Desired forward/backward torso lean angle during walking.
w foot pitch	Weight penalizing deviations in foot pitch orientation during optimization.
β post	General posture regularization weight encouraging nominal body posture.
w overall posture multiplier	Global scaling factor for all posture-related optimization terms.
w pelvis yaw	Weight penalizing pelvis yaw rotation to maintain heading alignment.
w pelvis roll	Weight penalizing pelvis roll motion for lateral stability.
β pelvis yaw	Regularization parameter controlling smooth pelvis yaw behavior.
β pelvis roll	Regularization parameter controlling smooth pelvis roll behavior.
w foot pitch	Weight regulating foot pitch motion and orientation changes.
right leg default posture	Nominal joint posture configuration for the right leg.
β subtalar	Weight controlling subtalar joint motion for foot inversion/eversion stability.
w xy feet	Penalty weight for horizontal foot placement and motion tracking.
w z feet	Penalty weight for vertical foot trajectory and motion regulation.
w s feet single	Foot-related optimization weight during the single-support phase of walking.
β torso	Weight regulating torso state motion and trajectory smoothness.
w acceleration torso xy	Weight penalizing horizontal torso accelerations for smoother motion.
w acceleration torso z	Weight penalizing vertical torso accelerations to reduce bouncing.
w_f CoM xy	Weight on horizontal CoM force generation during locomotion.
w_f CoM z	Weight on vertical CoM support force generation.
w_τ CoM xy	Weight penalizing COM torques around horizontal axes for rotational stability.
w_τ CoM z	Weight penalizing yaw torque around the CoM.
β square base forces	Regularization weight minimizing squared ground reaction/contact forces.
LLC eigenvalue	Eigenvalue tuning parameter for low-level controller (LLC) responsiveness and stability.
GRF phase in threshold	Threshold controlling smooth activation of ground reaction forces (GRF) during contact transitions.
contact force scale factor	Global scaling factor applied to computed contact forces.