Early Stopping Bayesian Optimization for Controller Tuning

Stenger, David; Scheurenberg, Dominik; Vallery, Heike; Trimpe, Sebastian

DOI
10.1109/CDC56724.2024.10886051

Publication date
2025

Document Version
Final published version

Published in
Proceedings of the IEEE 63rd Conference on Decision and Control, CDC 2024

2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Early Stopping Bayesian Optimization for Controller Tuning

David Stenger[1], Dominik Scheurenberg[2], Heike Vallery[2,3], and Sebastian Trimpe[1]

2024 IEEE 63rd Conference on Decision and Control (CDC) | 979-8-3503-1633-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/CDC56724.2024.10886051

***Abstract*— Manual tuning of performance-critical controller parameters can be tedious and sub-optimal. Bayesian Optimization (BO) is an increasingly popular practical alternative to automatically optimize controller parameters from few experiments. Standard BO practice is to evaluate the closed-loop performance of parameters proposed during optimization on an episode with a fixed length. However, fixed-length episodes can be wasteful. For example, continuing an episode where already the start shows undesirable behavior such as strong oscillations seems pointless. Therefore, we propose a BO method that stops an episode early if suboptimality becomes apparent before an episode is completed. Such early stopping results in partial observations of the controller's performance, which cannot directly be included in standard BO. We propose three heuristics to facilitate partially observed episodes in BO. Through five numerical and one hardware experiment, we demonstrate that early stopping BO can substantially reduce the time needed for optimization.**

I. INTRODUCTION

High-level control objectives, such as energy efficiency, passenger comfort, or product quality, cannot be directly encoded in many controllers. Therefore, appropriately tuning controller parameters is essential for achieving optimal closed-loop performance. However, manual tuning can be tedious and is potentially sub-optimal. In contrast, Bayesian Optimization (BO) [1] can automatically optimize controller parameters for almost arbitrary objectives and controller structures on real hardware. Examples include tuning linear quadratic regulators (LQR) in robotics [2], optimizing suspension controllers [3], and tuning of learning-based model predictive control (MPC) in autonomous racing [4].

BO has been shown to be more sample efficient, i.e., it can reach a better solution with fewer experiments than metaheuristics [5]. Furthermore, many advanced BO variants address important control challenges, such as safe tuning or online adaptation (see Sec. III).

At its core, BO leverages a probabilistic model of the analytically unknown objective function to suggest the next candidate controller, which is then evaluated in an experiment. The vast majority of BO for controller tuning conduct experiments of *fixed* episode length (see Sec. III). This common practice is, however, wasteful in terms of experimentation time, unnecessary wear, or energy consumption (e.g., [6]). Intuitively, an experimental episode can be stopped early if sub-optimality, e.g., oscillating behavior, of a given controller becomes apparent at the beginning of an episode.

We propose Early Stopping BO (ESBO) for time-integrated objectives such as energy consumption or mean-squared error (MSE). Based on past experiments, an application-independent rule is formulated to decide when to stop an episode. The early stopping of an episode results in a partial observation of the controller's performance. This partial observation cannot be used directly in the optimization because it would corrupt the probabilistic objective function model. To address this, we propose heuristics to facilitate the partial information. Deciding when to stop an episode early and how to facilitate the resulting partial information is a new problem class in BO for control (see Sec. III).

In summary, the main contributions of this paper are: (i) Formulate the ESBO problem for time-integrated objectives. (ii) Propose heuristic approaches to decide when to stop an episode and how to facilitate partially evaluated episodes in BO. (iii) Evaluate effectiveness in five simulation problems and one hardware experiment.

II. PROBLEM STATEMENT: EARLY STOPPING BO

In BO, the objective is to optimally choose a parameter vector $\boldsymbol{\theta}$ of the control law π that calculates the next control input u_{t+1} based on past inputs u and outputs y:

$$u_{t+1} = \pi(u_t, \dots, u_1, y_t, ..., y_1, \boldsymbol{\theta}). \tag{1}$$

The system dynamics are unknown to the optimization algorithm, and we make no assumptions about them. The cost function $J(\boldsymbol{\theta})$ captures the high-level control objective. Differently from vanilla BO, i.e., BO without early stopping (see, e.g., [1–4, 7] and Sec. IV-B), we assume that the objective is a sum of non-negative costs $j(u_t, y_t)$ generated at each time step t of an episode with length $T_{\max}$:

$$J(\boldsymbol{\theta}) = \sum_{t=1}^{T_{\max}} j(u_t, y_t), \quad j(u_t, y_t) \geq 0. \tag{2}$$

Examples of (2) are common control engineering metrics such as the Mean Squared/Absolute Error (MSE/MAE), rise time, LQR-costs, and the integral of time-multiplied absolute value of error (ITAE). Other metrics, e.g., settling time or metrics in the frequency domain, cannot be formulated this way.

The closed-loop behavior, i.e., u_t and y_t, depends on $\boldsymbol{\theta}$, so does the cost J. Hence, for a known feasible set Θ of

[1]Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Germany, e-mail: {david.stenger, trimpe}@dsme.rwth-aachen.de
[2]Institute of Automatic Control, RWTH Aachen University, Germany, e-mail: {d.scheurenberg,h.vallery}@irt.rwth-aachen.de
[3]Heike Vallery is also with the Department of BioMechanical Engineering, Delft University of Technology, and with the Department for Rehabilitation Medicine, Erasmus MC, Rotterdam, The Netherlands.

This work was partly funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the project EEMotion and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

dimension d, the tuning problem can be formalized as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta\subset\mathbb{R}^d} \quad J(\boldsymbol{\theta}). \tag{3}$$

The objective $J(\boldsymbol{\theta})$ is treated as a black-box function where the objective function's analytical structure, e.g., gradients, is unknown. Thus, to solve (3) we sequentially take data, i.e., select parameters $\boldsymbol{\theta}_k$ and perform an experiment to obtain a noisy evaluation J_k of $J(\boldsymbol{\theta}_k)$:

$$J_k = J(\boldsymbol{\theta}_k) + \epsilon_k. \tag{4}$$

In almost all BO controller tuning strategies, the experiment is performed over a fixed episode length $T_{\max}$, data $(u_{0,k}, ..., y_{T_{\max},k})$ is obtained and J_k is calculated directly according to (2). The additive noise ϵ may then represent measurement noise on y_t and is modeled as a Gaussian.

The objective of this work is to depart from the common assumption of a fixed evaluation length. Specifically, we consider the following problems:

(i) While performing the k-th evaluation, at which time T_k should the experiment be stopped?
(ii) If BO stopped an experiment early ($T_k < T_{\max}$), only data $(u_{0,k}, ..., y_{T_k,k})$ is available, and we cannot calculate J_k directly. How can BO process the incomplete data?

III. Related Work

Bayesian optimization has successfully been applied to various controller tuning contexts [2–5, 7] and other engineering domains [1, 8]. Furthermore, it can be applied to automatically optimize the hyperparameters of other learning-based control methods such as extremum-seeking control [7], learning-based MPC [4], and reinforcement learning (AutoRL) [9]. Due to its sample-efficiency [5], the automatic tuning of controllers with BO can be attributed to the micro-data branch of reinforcement learning [10]. BO was also applied to other control engineering domains, such as state estimation [11] or fault diagnosis [12].

BO was extended to effectively address numerous advanced challenges that arise in practical control engineering applications [13], e.g., contextual optimization [14], multiple objectives [15], constraints [16], time-varying optimization [17], multi-fidelity optimization [18], local optimization for high dimensional problems [19], and safe exploration [20]. Shortening episodes adaptively has been used for controller tuning of an exoskeleton [6]. However, a domain-specific stopping criterion only valid in this specific application was used in combination with vanilla BO. In the controller tuning context, existing BO methods *themselves* do not make active decisions about terminating an experiment. Thus early stopping is a new problem setting in BO for control.

An early-stopped evaluation leads to partial information about the objective. Learning with Crash Constraints (LCC) [5, 21] facilitates episodes that are not completed successfully, e.g., due to unstable behavior. However, in LCC, the decision to abort an episode is taken independently of BO, and BO only retrieves binary information on whether an evaluation was successful or not.

Freeze-thaw BO [22] has been proposed to abort the training procedure of machine learning (ML) models early. It relies on two assumptions that are not given in ESBO. First, a kernel is used that relies on an exponential decay of the training loss that is typical to ML training. The additive cost (2) does not have to follow this assumption. It does not even have to be differentiable, e.g., if the absolute error is used as a metric. Second, the model training can be resumed at any point. Freeze-thaw BO is extended in [23] by using Bayesian early stopping. However, still, an exponentially decaying kernel is used, which we cannot assume from (2).

Related problem formulations also occur in multi-task or robust BO. For example, [24] considers the problem formulation

$$\max_{\boldsymbol{\theta}\in\Theta\subset\mathbb{R}^d} \sum_{t\in\mathcal{T}} j(\boldsymbol{\theta}, t)p(t), \tag{5}$$

where $p(t)$ denotes the distribution of t. In [24], BO chooses which pairs $(\boldsymbol{\theta}, t)$ to evaluate at each iteration. This formulation is similar to (2) on the surface. However, the key difference is that ESBO cannot freely choose t. Instead, ESBO always evaluates time sequences starting with $t = 0$ up to $t = T_k$.

In summary, to the best of our knowledge, ESBO has not been considered for controller tuning, and related methods in other domains rely on different assumptions.

IV. Background

This section introduces Gaussian processes regression and vanilla BO without early stopping.

A. Gaussian Process Regression

In this paper, BO uses a Gaussian Process (GP) as a fast-to-evaluate probabilistic surrogate for the unknown scalar function $J(\boldsymbol{\theta})$. GPs are a class of non-parametric probabilistic regression models that are defined by a prior mean function

$$m(\boldsymbol{\theta}) = \mathbb{E}[J(\boldsymbol{\theta})] \tag{6}$$

and a covariance function or kernel

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \mathbb{E}[(J(\boldsymbol{\theta}) - m(\boldsymbol{\theta}))(J(\boldsymbol{\theta}') - m(\boldsymbol{\theta}')], \tag{7}$$

where $\mathbb{E}$ is the expectation. Here, we assume $m(\boldsymbol{\theta}) = 0$ because we do not have prior knowledge of the objective function landscape.

To approximate $J(\boldsymbol{\theta})$, we start with a GP prior that is updated with data

$$\mathcal{D}_k = \{(\boldsymbol{\theta}_1, J_1), ..., (\boldsymbol{\theta}_k, J_k)\}. \tag{8}$$

This data $\mathcal{D}_k$ is obtained by performing expensive experiments (4) corrupted by identically distributed Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. We denote the vector of noisy observations as $\boldsymbol{J}_k = [J_1, ..., J_k]^T$. The posterior mean μ_k and posterior variance σ_k^2 then are

$$\mu_k(\boldsymbol{\theta}) = \boldsymbol{k}_k(\boldsymbol{\theta})^T(\mathbf{K}_k + \sigma_n^2\mathbf{I}_k)^{-1}\boldsymbol{J}_k \tag{9}$$

$$\sigma_k^2(\boldsymbol{\theta}) = k(\boldsymbol{\theta}, \boldsymbol{\theta}) - \boldsymbol{k}_k(\boldsymbol{\theta})^T(\mathbf{K}_k + \sigma_n^2\mathbf{I}_k)^{-1}\boldsymbol{k}_k(\boldsymbol{\theta}). \tag{10}$$

Algorithm 1 Early Stopping BO

1: **Input**: randomly chosen initial parameters $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{k_{\text{init}}}$
2: $\mathcal{D}_k \leftarrow$ evaluate complete episodes with $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{k_{\text{init}}}$
3: $k \leftarrow k_{\text{init}}$
4: **repeat**
5: $\quad k \leftarrow k+1$
6: $\quad J_k^* \leftarrow$ identify current optimum from $\mathcal{D}_k$
7: $\quad \hat{\mathcal{D}}_k \leftarrow$ build virtual dataset using $\mathcal{D}_k$ ▷ cf. Sec. V-B
8: $\quad \mathcal{GP}_k \leftarrow$ fit GP model of $J(\boldsymbol{\theta})$ using $\hat{\mathcal{D}}_k$
9: $\quad \boldsymbol{\theta}_{k+1} = \text{argmax}_\Theta \alpha(\mathcal{GP}_k)$
10: $\quad (T_{k+1}, j_{k+1,1}, \dots, j_{k+1,T_{k+1}}) \leftarrow$ run episode with $\boldsymbol{\theta}_{k+1}$ and J_k^* ▷ cf. Alg. 2
11: $\quad \mathcal{D}_{k+1} \leftarrow \mathcal{D}_k \cup \{(\boldsymbol{\theta}_{k+1}, T_{k+1}, j_{k+1,1}, \dots, j_{k+1,T_{k+1}})\}$
12: **until** $k+1 \geq K$ or $\sum T_1 \cdots T_{k+1} \geq T_{\text{budget}}$
13: **return** $\boldsymbol{\theta}^*$

The gram matrix $\mathbf{K}_k \in \mathbb{R}^{k \times k}$ contains $[k(\boldsymbol{\theta}, \boldsymbol{\theta}')]_{\boldsymbol{\theta},\boldsymbol{\theta}' \in \mathcal{D}_k}$, the vector $\boldsymbol{k}_k(\boldsymbol{\theta}) = [k(\boldsymbol{\theta}_1, \boldsymbol{\theta}) ... k(\boldsymbol{\theta}_k, \boldsymbol{\theta})]$ describes the covariances between $\boldsymbol{\theta}$ and the observed data points $\boldsymbol{\theta}_1 \cdots \boldsymbol{\theta}_k$ and $\mathbf{I}_k$ is the $k \times k$ identity matrix. The GP model trained with data $\mathcal{D}_k$ is denoted as $\mathcal{GP}_k$. See [25] for more details.

B. Bayesian Optimization

Because evaluating $J(\boldsymbol{\theta})$ is costly, BO leverages all data obtained until iteration k to choose the next parametrization $\boldsymbol{\theta}_{k+1}$. After k_{init} random samples are evaluated, BO takes two steps to maximize the utility of the next experiment. First, a GP (see Sec. IV-A) is trained on all past observations to approximate $J(\boldsymbol{\theta})$. Second, this model is used in an acquisition function to balance exploration and exploitation. The acquisition function α uses the probabilistic GP predictions to calculate the utility of an experiment. It is maximized to find the next query:

$$\boldsymbol{\theta}_{k+1} = \text{argmax}_\Theta \alpha(\mathcal{GP}_k). \tag{11}$$

Approximately solving (11) is much easier than the original problem (3) because only the fast-to-evaluate GP model needs to be evaluated. This new parametrization is evaluated, new data is received, and the next iteration is started by again updating the GP model. This way, the GP model is iteratively refined around the optimum. We refer to [1, 8] for a detailed introduction to BO and to Sec. VI-A for implementation details.

V. Early Stopping BO

This section extends vanilla BO to ESBO. Sections V-A and V-B address the questions of when to abort an episode and how to facilitate partial observations (see problems (i) and (ii) of Sec. II). The expected strengths and weaknesses of the proposed approaches are discussed in Sec. V-C, and an illustrative example is given in Fig. 1. Algorithm 1 summarizes the proposed ESBO algorithm, and the changes to vanilla BO occur in lines 7 and 10.

A. When to stop an episode?

Algorithm 2 states how a parametrization is evaluated in ESBO. At each time step of the episode, input and output are retrieved, and the time step-specific cost is calculated.

Algorithm 2 Episode with Early Stopping

1: **Input**: parameter $\boldsymbol{\theta}_k$, maximum episode length, $T_{\max}$, and current best observation J_k^*
2: Assign parameters $\boldsymbol{\theta}_k$ to controller and start the episode
3: **for** $t = (1, 2, \cdots)$ **do**
4: $\quad (u_{t,k}, y_{t,k}) \leftarrow$ obtain inputs and outputs from process
5: $\quad j_{t,k} \leftarrow j(u_{t,k}, y_{t,k})$ calculate cost at timestep t
6: $\quad$ **if** $t == T_{\max}$ or $S(j_{1,k}, \dots, j_{t,k}, J_k^*)$ **then** ▷ cf. Sec. V-A
7: $\quad\quad$ **return** $(T_k = t, j_{1,k}, \dots, j_{t,k})$
8: $\quad$ **end if**
9: **end for**

We choose to stop an episode at time T_k if the parametrization cannot improve the current optimum, i.e., cannot return a smaller cost than the current optimum J_k^*. From now on, $j_{t,k}$ is used to denote a sample of $j(u_k, y_k)$. Because $j_{\cdot,\cdot} > 0$, we formulate the stopping decision rule S:

$$S(j_{1,k}, \dots, j_{t,k}, J_k^*) := \begin{cases} \text{true}, & \text{if} \quad \sum_{\tau=1}^{t} j_{\tau,k} \geq J_k^* \\ \text{false}, & \text{otherwise.} \end{cases} \tag{12}$$

After the episode is stopped, the partially obtained cost and the stopping time are returned to BO (Line 10, Alg. 1). If the maximum episode length $T_{\max}$ is reached, the episode is also stopped, and a new best solution is found.

B. How to facilitate partial observations?

If an episode is aborted early ($T_k < T_{\max}$), only partial knowledge $J_{\text{part},k} = \sum_{t=1}^{T_k} j_{t,k}$ about the objective is available. We cannot directly use $J_{\text{part},k}$ in the GP model for $J(\boldsymbol{\theta})$ because it would underestimate $J(\boldsymbol{\theta})$. Therefore, we create virtual data points that (i) guide the optimization away from the sub-optimal regions while (ii) not contradicting the smoothness assumptions of the GP and (iii) leverage the partial information. Below, three heuristics are proposed. Note that all heuristics recalculate all virtual data points at each iteration.

a) ESBO-C: This approach treats partial observations as crashes (cf. Sec. III). Virtual data points are generated for all incomplete episodes l by using pessimistic GP predictions as proposed in [5, 13]:

$$\hat{J}_l = \min\{\max\{\bar{\mu}_J(\boldsymbol{\theta}_l), J_k^*\} + 3\bar{\sigma}_J(\boldsymbol{\theta}_l), J_{\max,k}\}, \tag{13}$$

where $\bar{\mu}_J$ and $\bar{\sigma}_J$ are the predicted mean and standard deviation from a GP built with only data from complete episodes. The objectives of the best and worst complete episodes obtained so far are denoted by J_k^* and $J_{\max,k}$. The lower bound is enforced to ensure that the virtual data point is worse than the current optimum. The upper bound constrains it in case of large predictive uncertainties.

Because the virtual data points use probabilistic GP predictions, they ensure a smooth transition of the predictions between complete and partially observed costs. The pessimistic realization ensures that the optimization is driven away from the partial observation (cf. [13, Fig. 4.9]).

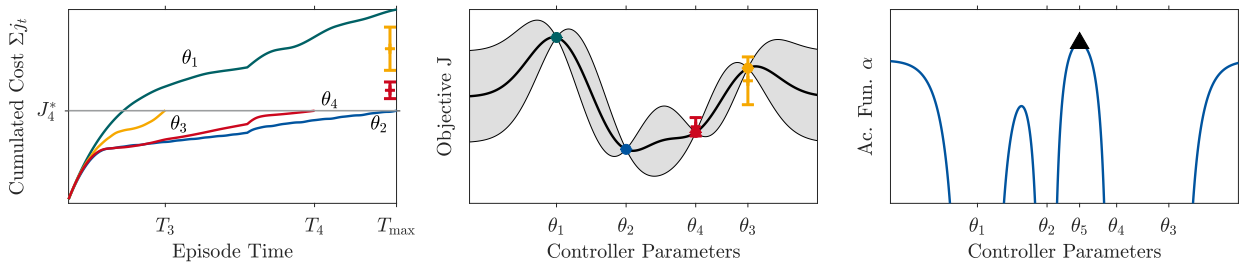

Fig. 1. Illustrative example of ESBO-GP: The first and the second samples θ_1 and θ_2 were evaluated for the whole episode. The second sample θ_2 achieved an improvement. The evaluation of the third and fourth samples θ_3 and θ_4 was stopped early, because already after T_3 and T_4, an improvement was ruled out by the early stopping rule (see Sec. V-A). The objective function values of θ_3 and θ_4 are sampled from a probabilistic estimate and included in the GP (see Sec. V-B). The GP model of the objective function uses the observed total cost for the complete evaluations. By using the partial information in the GP, the acquisition function is smaller in areas where partial observations have been made, and the parametrization for the next episode is found by maximizing the acquisition function.

b) ESBO-TR: The main idea of the time reformulation (ESBO-TR) approach is that with an objective function structure of (2), the time T_m^* needed until the cumulative cost of parametrization m reaches the current optimum

$$\sum_{t=1}^{T_m^*} j_{t,m} \leq J_k^*. \tag{14}$$

is a notion of fitness in itself. Thus, searching for a parametrization that would increase the episode length, i.e., that would not be aborted under the current optimum, coincides with searching for parameters with a good objective function value. At each iteration k for all observations $m \in \{1, \ldots, k\}$, we find the time T_m^*, after which the cumulative cost exceeds the current observed optimum J_k^* and assign it as the virtual objective function value: $\hat{J}_m = -T_m^*$. Thus, for the current optimum, we assign $-T_{\max}^*$. The minus sign is needed because we maximize episode length to minimize cost.

c) ESBO-GP: The main idea is to probabilistically estimate the unobserved portion, T_l to $T_{\max}$, of partially observed episode l with separate GPs (see Fig. 1).

First, the time period of an episode is split into different sections, the first section starting at T_1^{sort} and ending at T_2^{sort}:

$$(T_1^{\text{sort}}, \ldots, T_{N_{\text{GP}}+1}^{\text{sort}}) = \text{unique}(T_1, \cdots T_k).$$

The unique operator sorts the previously obtained episode lengths and removes duplicates. In the example of Fig. 1, the unique episode lengths are $(T_3, T_4, T_{\max})$. Next, N_{GP} GPs are built to model the cumulated cost $j_{\text{sec},\cdot}$ for each section between the unique episode lengths:

$$\begin{aligned} j_{\text{sec},1}(\boldsymbol{\theta}) &= \textstyle\sum_{t=T_1^{\text{sort}}}^{T_2^{\text{sort}}} j(u_t, y_t) \sim \mathcal{GP}_1, \\ &\cdots \\ j_{\text{sec},N_{\text{GP}}}(\boldsymbol{\theta}) &= \textstyle\sum_{t=T_{N_{\text{GP}}}^{\text{sort}}}^{T_{N_{\text{GP}}+1}^{\text{sort}}} j(u_t, y_t) \sim \mathcal{GP}_{N_{\text{GP}}}. \end{aligned} \tag{15}$$

The GPs are trained using all data available for the respective sections. In the example of Fig. 1, two additional GPs are trained, one from T_3 to T_4 and one from T_4 to $T_{\max}$. Using the GP models, we make probabilistic predictions for each section q and partially evaluated parametrization l:

$$j_{\text{sec},q}(\boldsymbol{\theta}_l) \sim \mathcal{N}(\mu_q(\boldsymbol{\theta}_l), \sigma_q^2(\boldsymbol{\theta}_l)), \forall q \in \{1, \cdots N_{\text{GP}}\}. \tag{16}$$

These are then added to the partial evaluation to generate the probabilistic predictions for the full cost:

$$\begin{aligned} \tilde{J}_l &\sim \mathcal{N}(\mu_{\tilde{J}_l}, \sigma_{\tilde{J}_l}^2), \quad \text{with} \\ \mu_{\tilde{J}_l} &= \sum_{t=1}^{T_l} j_{t,l} + \mu_p(\boldsymbol{\theta}_l) + \cdots + \mu_{N_{\text{GP}}}(\boldsymbol{\theta}_l) \\ \sigma_{\tilde{J}_l}^2 &= \sigma_p^2(\boldsymbol{\theta}_l) + \cdots + \sigma_{N_{\text{GP}}}^2(\boldsymbol{\theta}_l). \end{aligned} \tag{17}$$

We cumulate the probabilistic estimates starting with GP model p that models the section starting at T_l.

The virtual data point $\hat{J}_l$ is sampled randomly at each iteration from the distribution of $\mathcal{N}(\mu_{\tilde{J}_l}, \sigma_{\tilde{J}_l}^2)$. Additionally, we bound $\hat{J}_l$ to be larger than the known partially obtained cost $\sum_{t=1}^{T_l} j_{t,l}$.

We also tried using the probabilistic estimate $\tilde{J}_l$ directly in the GP for $J(\boldsymbol{\theta})$ as a virtual data point with known heteroscedastic noise or model $j(\boldsymbol{\theta}, t)$ directly as a GP similarly to [22]. These two options were not pursued further because preliminary results were not promising. The former option can lead to a large predictive uncertainty at early stopped parametrizations resulting in a non-zero acquisition function. However, this uncertainty cannot be reduced by resampling the same location.

C. Expected Strengths and Weaknesses

In BO, the evaluation of a parametrization serves two main purposes. First, we want to determine if the parametrization leads to a better solution than previously found. This is ensured by (12). Secondly, completing an episode may be valuable in improving the GP model to make better-informed future parameter choices, especially if substantial time is needed to reset the episode. The latter is not considered here. In contrast, an episode could be aborted even earlier by using a probabilistic model of the future cost of the remaining episode [22] (see also Bayesian early stopping [23]). This may be especially useful in case the majority of the cost is generated late in the episode and is also not considered here.

It is expected that using more data from partial observations is beneficial for the progress of the optimization. ESBO-C only uses the binary information on whether the episode was stopped early. ESBO-TR uses information until the respective episodes would have been stopped under the current optimum, and ESBO-GP leverages all available data from the partial observations.

Noisy optimization may be challenging for ESBO-TR because we cannot directly model the observation noise on $J_k(\boldsymbol{\theta})$ by applying the time reformulation.

VI. Empirical Evaluation

After introducing hyperparameters and evaluation metrics, this section evaluates the proposed ESBO variants on five deterministic stimulation and one hardware controller tuning tasks. Our experiments reveal the following key findings:

- In simulations, ESBO-C and ESBO-GP perform similarly and outperform ESBO-TR, vanilla BO, and random search baselines.
- In simulations, ESBO-GP speeds up optimization by up to 48% simulation time steps in comparison to vanilla BO without compromising the quality of the solution at maximum budget.
- In the hardware experiment, ESBO-GP speeds up optimization by up to 35% experimentation time and finds final solutions comparable to vanilla BO.

Our implementation used in the experiments is available[1].

A. Implementation and Hyperparameters

The simulation and hardware experiments aim at determinining the of-the-box-performance of the proposed algorithms. Therefore, none of the BO hyperparameters were adjusted to the test cases, and all experiments use identical hyperparameters.

Max-value entropy search (MES) [26] is used as the acquisition function. MES does not rely on additional decisive parameters and applies to noisy and deterministic optimization. The acquisition function is multimodal. Therefore, the acquisition function maximization (11) is performed by a combination of random sampling and gradient descent.

For GP modeling, the input $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_k$ is transformed to the unit cube, and the output $\hat{J}_1, \dots, \hat{J}_k$ is normalized to zero mean and a standard deviation of one. We use a zero prior mean and an anisotropic squared-exponential kernel for the transformed data. GP-hyperparameters (signal strength, length scales, and observation noise for the hardware experiments) are optimized in each iteration by maximizing the log-likelihood. Additionally, length scales are bounded such that a correlation of 0.1 is possible over at most half the domain and at least over 1% of the domain. This ensures that length scales are within a similar order of magnitude as the optimization domain.

In the hardware experiment and three of the five simulation experiments, some parameterizations lead to crashes due to instability or tank overflow (cf. crash constraints Sec. III).

TABLE I
Simulative controller tuning tasks with problem dimension d

No.	Controller	Plant	d	Objective	Crashes? (VI-A)
1	Bang-Bang	Boiler	1	Energy & Comfort	No
2	PI	Three Tank [27]	2	MSE	No
3	PID	PT-2 w. Dead-Time	3	ITAE	Yes
4	State Feedback	Cart Pole	4	LQR-Costs	Yes
5	MPC	Three tank [27]	5	MAE	Yes

Here, we use BO-VDP [13] to generate virtual data points according to (13) for the crashed evaluations. Our implementation[1] is based on the GPML toolbox[2].

B. Evaluation Metrics

We use different test cases and seeds, i.e., initial samples, to empirically evaluate ESBO's performance. Simple regret and rank are used to quantify the optimization progress. The simple regret r_k after k evaluations is calculated as the difference $r_k = J_k^* - J^*$ between the best evaluation found so far J_k^* and the overall optimum J^*. We take the best evaluation across all algorithms and seeds for J^*. To make it comparable across different algorithms, we scale the regret such that a scaled regret of 1 corresponds to the median performance of random search at maximum budget. The scaled regret is then averaged over all test cases and seeds. [5]

We sort the regrets of the different evaluated algorithms to calculate the rank. The algorithm with the lowest regret gets ranked 1, the second best gets ranked 2, and so on. The average rank is calculated by simply averaging over test cases and seeds. [5]

The motivation of ESBO is to reduce total interaction time, i.e., simulation time steps or hardware experimentation time. Therefore, results are reported with respect to interaction time and evaluations.

C. Simulation Study

Setup: Table I summarizes the simulation test cases. Five controllers ranging from a simple bang-bang controller to a linear MPC are optimized. Plant models vary from a synthetic second-order system (PT-2) with dead time to a more complex non-linear three-tank system.

To evaluate the effectiveness of ESBO-C, ESBO-TR, and ESBO-GP, we use three baselines: (i) Random Sampling with early stopping (ESRS), i.e., choosing parameters $\boldsymbol{\theta}$ at random and using the early stopping rule (12), (ii) Random Sampling (RS), i.e., same as ESRS but without early stopping, (iii) BO without early stopping (BO).

[1] https://github.com/Data-Science-in-Mechanical-Engineering/ESBO

[2] http://gaussianprocess.org/gpml/code/matlab/doc/

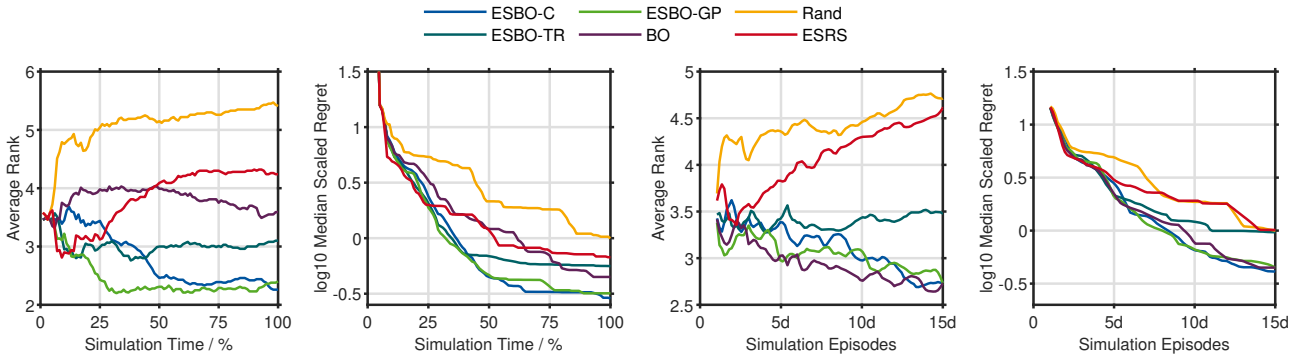

Fig. 2. Results of the simulation study. For an explanation of the metrics see Sec. VI-B.

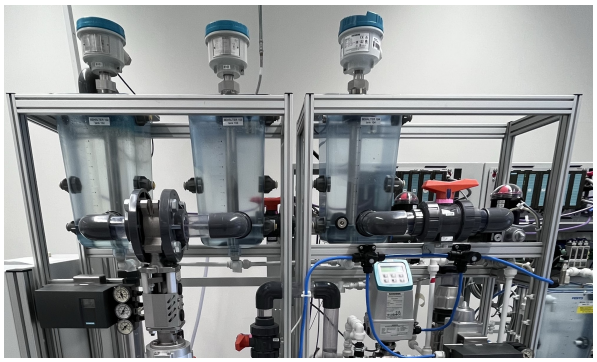

Fig. 3. Three tank hardware test bed at the model factory of *Institute of Automatic Control, RWTH Aachen University*.

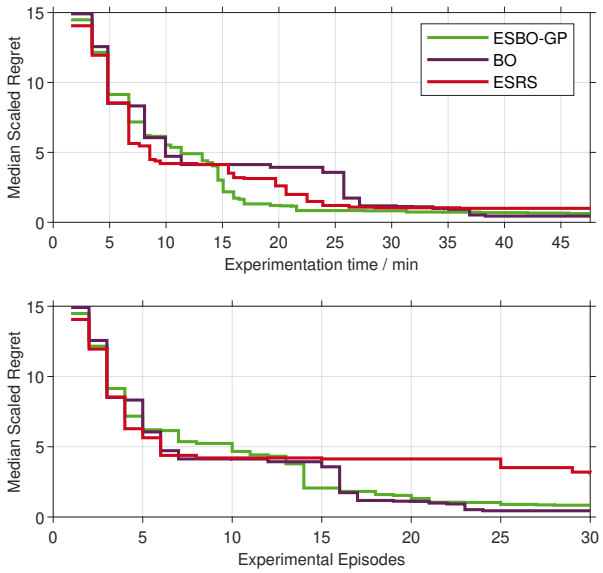

Fig. 4. The two plots represent the same hardware experiments, showing median scaled regret over experimentation time (top) and number of episodes (bottom). For an explanation of the metrics see Sec. VI-B.

Each optimization is run with 10 different random seeds. The initial sample size is set to the problem dimensionality $k_{\text{init}} = \max\{d, 2\}$. The optimization is terminated if $T_{\text{budget}} = 15 \cdot d \cdot T_{\max}$ time steps or $K = 45 \cdot d$ objective function queries are exceeded. Box constraints $\boldsymbol{\theta}_{\min} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_{\max}$ define the domain Θ.

Results: Figure 2 shows the simulation results averaged over all test cases. All three heuristics reach better results faster than vanilla BO and RS. The simulation time needed to achieve the median final performance of ESRS is reduced by 48% from 62% to 32% percent using ESBO-GP instead of BO. The performance after T_{budget} is best using ESBO-C and ESBO-GP. The ESBO variants need comparably many evaluations as vanilla BO and the RS variants are not competitive.

D. Hardware Experiments

Setup: We evaluate the practical applicability of the ESBO method on a three-tank hardware test bed (cf. Fig. 3). A detailed description of the test bed is given in [27].

A PI controller tracks a reference step in the desired water level of the third tank. Thus, the proportional and integral gains of the PI controllers are optimized in the log domain. The objective J is the mean squared tracking error plus a penalty on large pump actuation.

Because it performed most consistently in simulation ESBO-GP is evaluated on the hardware test bed with BO and ESRS as benchmarks. All algorithms are run with eight randomly selected starting parameterizations each. The same budget and metrics as in Sec. VI-C are used, with the exception of an initial sample size of $k_{\text{init}} = 3$. Note that in contrast to the simulation study, noise is now present. Therefore, BO and ESBO-GP include the observation noise σ_n^2 in the hyperparameter optimization.

Results: Figure 4 shows the optimization progress. ESBO-GP reaches better solutions at around 15 to 26 minutes and achieves a comparable final solution. The experimentation time needed to achieve the median final performance of ESRS is reduced by 35% from 34 to 22 minutes using ESBO-GP.

Both BO variants outperform random search when considering progress w.r.t. episodes (queries).

We evaluated the statistical significance of the results using a rank-sum test at 5% significance level. After around 30 minutes, the performance difference is statistically insignificant, while ESRS and BO perform significantly worse than

ESBO-GP at around 15 to 26 minutes. Note that the slight differences in initial performance after experiment one occur due to process noise.

E. Discussion

In addition to reducing interaction time, the ESBO variants and BO exhibit similar query efficiency. This indicates that the ESBO heuristics proposed to generate the virtual data points are sufficient to enable optimization although samples are less informative in ESBO. ESBO-GP is the best-performing heuristic possibly because it uses all information obtained from the partial evaluations. Early stopping benefits random search substantially, making it a competitive alternative to vanilla BO. Random search is outperformed consistently, highlighting that the hardware and simulation test cases chosen are not trivial to solve.

VII. Conclusion

We introduced ESBO to automate controller tuning, a setting where BO stops the closed-loop evaluation before nominal completion. We proposed heuristic approaches to address ESBO's key challenges: (i) decide when to stop an episode and (ii) enable the resulting partial evaluations.

The proposed methods were evaluated in five diverse simulation tasks and one hardware experiment. Overall, results highlight the potential of ESBO to speed up automated controller tuning without sacrificing final solution quality. The developed heuristics may serve as baselines for more principled early stopping BO approaches for controller tuning in future research.

Acknowledgments

We thank Alexander von Rohr, Johanna Menn, and Paul Brunzema for many helpful comments, suggestions, and discussions.

References

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

[2] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 270–277.

[3] G. Savaia, Y. Sohn, S. Formentin, G. Panzani, M. Corno, and S. M. Savaresi, "Experimental automatic calibration of a semi-active suspension controller via bayesian optimization," *Control Engineering Practice*, vol. 112, p. 104 826, 2021.

[4] L. P. Fröhlich, C. Küttel, E. Arcari, L. Hewing, M. N. Zeilinger, and A. Carron, "Contextual tuning of model predictive control for autonomous racing," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 10 555–10 562.

[5] D. Stenger and D. Abel, *Benchmark of bayesian optimization and metaheuristics for control engineering tuning problems with crash constraints*, 2022. [Online]. Available: https://arxiv.org/abs/2211.02571.

[6] M. Kim, C. Liu, J. Kim, S. Lee, A. Meguid, C. J. Walsh, and S. Kuindersma, "Bayesian optimization of soft exosuits using a metabolic estimator stopping process," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9173–9179.

[7] A. Chakrabarty, D. J. Burns, M. Guay, and C. R. Laughman, "Extremum seeking controller tuning for heat pump optimization using failure-robust bayesian optimization," *Journal of Process Control*, vol. 120, pp. 86–96, 2022.

[8] 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, no. 1, pp. 148–175, 2016.

[9] J. Parker-Holder, R. Rajan, X. Song, A. Biedenkapp, Y. Miao, T. Eimer, B. Zhang, V. Nguyen, R. Calandra, A. Faust, *et al.*, "Automated reinforcement learning (autorl): A survey and open problems," *Journal of Artificial Intelligence Research*, vol. 74, pp. 517–568, 2022.

[10] K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J.-B. Mouret, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 328–347, 2020.

[11] M. Nitsch, D. Stenger, and D. Abel, "Automated tuning of nonlinear kalman filters for optimal trajectory tracking performance of auvs," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 11 608–11 614, 2023, 22nd IFAC World Congress.

[12] J. Marzat, H. Piet-Lahanier, and E. Walter, "Min-max hyperparameter tuning, with application to fault detection," *18th IFAC World Congress*, 2011.

[13] D. Stenger, "Automatic tuning of control engineering algorithms with Bayesian optimization," Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2023. [Online]. Available: https://publications.rwth-aachen.de/record/971795.

[14] M. Fiducioso, S. Curi, B. Schumacher, M. Gwerder, and A. Krause, "Safe contextual Bayesian optimization for sustainable room temperature PID control tuning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, (IJCAI-19)*, Jul. 2019, pp. 5850–5856.

[15] G. Makrygiorgos, A. D. Bonzanini, V. Miller, and A. Mesbah, "Performance-oriented model learning for control via multi-objective Bayesian optimization," *Computers & Chemical Engineering*, vol. 162, p. 107 770, 2022.

[16] M. Khosravi, C. König, M. Maier, R. S. Smith, J. Lygeros, and A. Rupenyan, "Safety-aware cascade controller tuning using constrained Bayesian optimization," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 2, pp. 2128–2138, 2023.

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

[18] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 1557–1563.

[19] A. von Rohr, D. Stenger, D. Scheurenberg, and S. Trimpe, "Local bayesian optimization for controller tuning with crash constraints," *at–Automatisierungstechnik*, vol. 72, no. 4, pp. 281–292, 2024.

[20] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *Machine Learning*, vol. 112, pp. 3713–3747, 2021.

[21] A. Marco, D. Baumann, M. Khadiv, P. Hennig, L. Righetti, and S. Trimpe, "Robot learning with crash constraints," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1439–1446, 2021.

[22] K. Swersky, J. Snoek, and R. P. Adams, *Freeze-thaw bayesian optimization*, 2014. [Online]. Available: https://arxiv.org/abs/1406.3896.

[23] Z. Dai, H. Yu, B. K. H. Low, and P. Jaillet, "Bayesian optimization meets Bayesian optimal stopping," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, PMLR, 2019, pp. 1496–1506.

[24] S. Toscano-Palmerin and P. I. Frazier, "Bayesian optimization with expensive integrands," *SIAM Journal on Optimization*, vol. 32, no. 2, pp. 417–444, 2022.

[25] C. E. Rasmussen and Williams C. K. I., "Gaussian Processes for Machine Learning," 2006.

[26] Z. Wang and S. Jegelka, "Max-value entropy search for efficient bayesian optimization," in *International Conference on Machine Learning*, PMLR, 2017, pp. 3627–3635.

[27] D. Scheurenberg, S. Stemmler, and D. Abel, "Evaluation of data enhanced model predictive control for a coupled tank system," in *2023 IEEE Conference on Control Technology and Applications (CCTA)*, 2023, pp. 79–84.