Methods

Alexander von Rohr*, David Stenger, Dominik Scheurenberg and Sebastian Trimpe

Local Bayesian optimization for controller tuning with crash constraints

Lokale Bayes'sche Optimierung für die Reglereinstellung mit Absturzbeschränkungen

https://doi.org/10.1515/auto-2023-0181
Received October 1, 2023; accepted February 23, 2024

Abstract: Controller tuning is crucial for closed-loop performance but often involves manual adjustments. Although Bayesian optimization (BO) has been established as a data-efficient method for automated tuning, applying it to large and high-dimensional search spaces remains challenging. We extend a recently proposed local variant of BO to include crash constraints, where the controller can only be successfully evaluated in an *a-priori* unknown feasible region. We demonstrate the efficiency of the proposed method through simulations and hardware experiments. Our findings showcase the potential of local BO to enhance controller performance and reduce the time and resources necessary for tuning.

Keywords: controller tuning; Bayesian optimization; learning-based control

Zusammenfassung: Eine korrekte Reglerparametrierung ist entscheidend für die Güte des geschlossenen Regelkreis, erfordert jedoch häufig manuelle Anpassungen. Obwohl sich Bayes'sche Optimierung (BO) als dateneffiziente Methode für die automatische Einstellung etabliert hat, bleibt ihre Anwendbarkeit auf große und hochdimensionale Suchräume eine Herausforderung. Wir erweitern eine kürzlich vorgeschlagene lokale Variante von BO um Absturzbeschränkungen, bei denen der Regler nur in einem *a-priori* unbekannten Bereich erfolgreich evaluiert werden kann. Wir demonstrieren die Effizienz der vorgeschlagenen Methode durch Simulationen und Hardwareexperimente. Unsere Ergebnisse zeigen das Potenzial von lokaler BO, die Regelgüte zu verbessern und dabei sowohl Zeit als auch Ressourcen zu sparen.

Schlagwörter: Reglereinstellung; Bayes'sche Optimierung; lernende Regler

***Corresponding author: Alexander von Rohr**, Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Aachen, Germany, E-mail: vonrohr@dsme.rwth-aachen.de. https://orcid.org/0000-0002-0005-0310
David Stenger and Dominik Scheurenberg, Institute of Automatic Control, RWTH Aachen University, Aachen, Germany, E-mail: d.stenger@irt.rwth-aachen.de (D. Stenger), d.scheurenberg@irt.rwth-aachen.de (D. Scheurenberg). https://orcid.org/0000-0003-3747-4499 (D. Stenger). https://orcid.org/0000-0002-1044-5631 (D. Scheurenberg)
Sebastian Trimpe, Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Aachen, Germany, E-mail: trimpe@dsme.rwth-aachen.de. https://orcid.org/0000-0002-2785-2487

1 Introduction

Most control algorithms involve user-defined parameters that determine the closed-loop behavior. Examples include controller gains for PID-controllers and stage and terminal costs in model predictive control (MPC). Inadequate choices for these parameters often lead to performance issues [1]. Controller tuning is the process of adjusting parameters to meet specified performance requirements for a given control task. Evaluation of the performance requires running experiments in either simulation or on hardware. Although analytical solutions for optimal parameters exist in some cases, for instance, for the linear quadratic regulator (LQR) and linear quadratic integral (LQI) control, practical applications often require adjustments of the weighting matrices to ensure the closed-loop satisfies performance requirements that are not captured in those cost functions or to counteract modeling inaccuracies.

Automation presents a promising solution to improve control performance during commissioning and in response to changes in operating conditions. Automated controller tuning aims to identify effective controllers by utilizing prior knowledge about the plant and data collected during its operation. An emerging approach in controller tuning is Bayesian Optimization (BO), which is particularly well-suited for this purpose due to its data-efficiency [2]–[4]. The controller tuning loop with BO is illustrated in Figure 1.

A well-known limitation of BO in a practical controller tuning setting is its dependence on the dimension

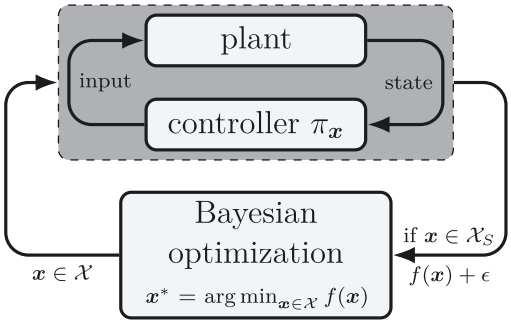


Figure 1: The controller tuning process with BO. The objective f is evaluated in closed-loop. The controller $\pi_{\boldsymbol{x}}$ has tuning parameters $\boldsymbol{x} \in \mathcal{X}$ and BO searches for the optimal parameterization. No function value is available if an experiment crashes, $\boldsymbol{x} \notin \mathcal{X}_S \subseteq \mathcal{X}$.

of the search domain. Controller tuning involves evaluating the closed-loop performance through experiments, each of which can take several minutes. Consequently, conducting more than a few hundred evaluations can be impractical. Therefore, an automated tuning method should reliably identify effective controllers within a few evaluations. This requirement restricts BO's applicability to low-dimensional problems and small search domains.

Practical controller tuning can also introduce an additional layer of complexity in the form of crash constraints [5]. If control performance is evaluated by observing the closed-loop system, it may exhibit unsafe or undesired behavior, which requires terminating the experiment early. This often means that successful and crashed evaluations cannot be meaningfully expressed in the same metric. Crash constraints are common when tuning controllers for complex systems, such as in robotics and have already been considered in the first works on BO for controller tuning (e.g., [6], [7]). The ubiquity of crash constraints motivated specialized learning methods such as Marco et al. [8].

In previous work, we proposed Gradient Information with BO (GIBO) [9], a local variant of BO. Starting from an initial parametrization, GIBO uses a small number of evaluations to learn a descent direction to update the parameters in an improvement step. Empirical studies show that GIBO and its variants [10], [11] are more data-efficient and outperform global BO on synthetic benchmarks. However, the GIBO algorithm has not yet been applied to practical control problems. In this article, we revisit GIBO and investigate its applicability for practical controller tuning under crash constraints through simulation and hardware experiments.

The benefits of local search for controller tuning include higher data-efficiency for large and high-dimensional problems, continuous improvement, and local exploration. As an additional benefit, control engineers and other algorithm users may find local search more intuitive due to smaller updates, making these algorithms easier to understand and deploy. Nevertheless, local optimization is sensitive to the initial parameterization and may converge to sub-optimal local minima. Fortunately, prior work indicates that controller tuning problems frequently have a unique minimum [4].

1.1 Problem statement

The controller tuning problem is defined as optimizing an objective function which maps control algorithm parameters to the performance of the closed-loop system

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}_S} f(\boldsymbol{x}), \tag{1}$$

where $\mathcal{X}_S \subseteq \mathcal{X} \subset \mathbb{R}^d$ is the feasible region of the search space $\mathcal{X}$ and d is its dimensionality. The crash region is denoted as $\mathcal{X}_C = \mathcal{X} \backslash \mathcal{X}_S$. The objective f is a costly black-box function; that is, evaluations are required to obtain its function value, and these evaluations are resource-intensive. Note that the feasible region $\mathcal{X}_S$ and, therefore, the crash region $\mathcal{X}_C$ are unknown, and f is undefined outside of it. Evaluating f outside of $\mathcal{X}_S$ remains costly but will not yield an objective value. This problem formulation requires that there is an experimental procedure in place to evaluate performance and recover from a crashed evaluation. These procedures may be fully automated but can also require human interventions.

We assume we can collect data of the form $(\boldsymbol{x}, y)$ for any $\boldsymbol{x} \in \mathcal{X}_S$ with

$$y = f(\boldsymbol{x}) + \epsilon,$$

where $\epsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$ is independent and identically distributed Gaussian noise. We denote $\mathcal{D} = (X, \boldsymbol{y})$ as the dataset with $|\mathcal{D}| := N$ and

$$X := \begin{bmatrix} x_1, \\ \vdots \\ x_N \end{bmatrix} \boldsymbol{y} := \begin{bmatrix} y_1, \\ \vdots \\ y_N \end{bmatrix}.$$

No direct assumptions are made regarding the controlled system or control algorithm. However, we assume the objective function is a sample from a mean-square differentiable Gaussian process (GP). This assumption enables us to learn a local gradient and determine a search direction that enhances closed-loop behavior. It is worth noting that this type of regularity assumption is a standard practice in BO [12]. For practical purposes, it is common to select compact and convex sets as the search domain $\mathcal{X}$.

Assumption 1. *The performance function f is a sample from a Gaussian process with $p(f) = \mathcal{GP}(f; \mu, k)$, whose*

mean function $\mu: \mathcal{X} \to \mathbb{R}$ is at least once differentiable and whose covariance function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is at least twice differentiable.

1.2 Contributions

We propose GIBO with virtual data points (VDP-GIBO), a general and data-efficient optimization algorithm for controller tuning under crash constraints. The proposed method is based on prior work on local BO [9], which learns the gradient of the control objective with respect to the tuning parameters from noisy closed-loop performance evaluations. Crash constraints are addressed by introducing virtual data points [4], which guide the optimization away from the infeasible region. The proposed method is evaluated on a simulated coupled tank system with popular control algorithms, namely PI control, LQI and MPC. Additionally, we validate our method by tuning a PI controller on hardware.

2 Related work

This section is an overview of different controller tuning applications using BO. BO treats the tuning task as a black-box optimization problem. It is not restricted to a special system or objective function class. Therefore, it has been used to automatically optimize the parameters of different controller structures such as LQR [7], MPC [13], and PID [14]. In addition to controllers, BO has been applied to other control engineering algorithms such as Kalman filter [15] and fault diagnosis [16]. BO can be applied to complex hierarchical controller structures and the interaction between controllers and filters [17], [18].

The problem statement in (1) is a single-objective formulation with crash constraints. However, in practical controller tuning problems, more complex formulations may be required to capture the tuning task fully. With its various extensions, BO offers a versatile toolkit to address those issues. In contextual optimization [19], parameters are optimized as a function of an operating condition. Optimization with unknown constraints (e.g., [20]) restricts the set of feasible solutions. In contrast to the setting herein, safe BO tries to stay within those bounds also *during* optimization [21]. Pareto optimization simultaneously considers multiple objectives [22], [23]. Related topics include robust optimization [24], [25], preference-based tuning [26], and time-varying optimization problems [27].

Approaches to address crash constraints in controller tuning include assigning a fixed penalty (e.g., [6]) or using data obtained before the crash (e.g., [7]). However, it may require substantial domain knowledge to design the penalty. A probabilistic classifier in combination with constrained BO (e.g., [5], [28]) can also be used to address the issue. However, this may result in tedious tuning of the additional hyperparameters of the classifier. Marco et al. [8] propose a combined GP model for constrained optimization with crash constraint and apply it to a controller tuning task on a quadroped robot. This approach requires modifying the acquisition function to incorporate the separate model of the constraints.

Herein, we address crash constraints using BO with virtual data points (VDP-BO). The upside of VDP-BO is that it only modifies the GP modeling step of BO by introducing virtual observation. Therefore, the acquisition function step remains unchanged, and VDP-BO can be easily incorporated with different BO flavors. VDP-BO was introduced for single-objective optimization in [4], and applied to constrained [18] and multi-objective optimization [22].

3 Preliminaries

This section introduces Gaussian Processes (GPs) and their derivatives, along with a strategy for minimizing the posterior variance of gradient estimates. For an introduction to GPs and BO, we refer to Garnett [29].

3.1 Gaussian process derivatives

We utilize a GP to model the expectation and the uncertainty of the objective's gradient. This model guides the optimization procedure to improve the closed-loop performance quickly. Given the prior from Assumption 1 and a dataset of closed-loop performance observations $\mathcal{D}$, their joint distribution is

$$p(f, y) = \mathcal{GP}\left(\begin{bmatrix} f \\ \boldsymbol{y} \end{bmatrix}; \begin{bmatrix} \mu \\ \boldsymbol{m} \end{bmatrix}, \begin{bmatrix} k & \kappa^\top \\ \kappa & C \end{bmatrix}\right),$$

where $\boldsymbol{m} = \mu(X)$, $C = k(X, X) + \sigma_n^2 I$, and $\kappa = k(\cdot, X)$. Mean and covariances are given by μ and k. The posterior distribution at location $\boldsymbol{x}_*$ is (cf. [30])

$$p(f(\boldsymbol{x}_*) \mid \mathcal{D}) = \mathcal{N}\big(f(\boldsymbol{x}_*); \mu_D(\boldsymbol{x}_*), k_D(\boldsymbol{x}_*)\big),$$

where

$$\mu_D(\boldsymbol{x}_*) = \mu(\boldsymbol{x}_*) + \kappa(\boldsymbol{x}_*)^\top C^{-1}(\boldsymbol{y} - \boldsymbol{m})$$
$$k_D(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \kappa(\boldsymbol{x}_*)^\top C^{-1}\kappa(\boldsymbol{x}_*).$$

Analogously, the joint distribution between observations and the functions derivative is

$$p(\nabla f, y) = \mathcal{GP}\left(\begin{bmatrix}\nabla f \\ y\end{bmatrix}; \begin{bmatrix}\nabla \mu \\ \boldsymbol{m}\end{bmatrix}, \begin{bmatrix}\nabla k \nabla^\top & (\nabla \kappa)^\top \\ \nabla \kappa & C\end{bmatrix}\right),$$

where ∇ is the differential operator and ∇ placed behind k takes the derivative w.r.t. the second input. The posterior of the derivative at $\boldsymbol{x}_*$ is

$$p(\nabla f(\boldsymbol{x}_*) \mid \mathcal{D}) = \mathcal{N}\left(\nabla f(\boldsymbol{x}_*); \nabla \mu_{\mathcal{D}}(\boldsymbol{x}_*), \nabla k_{\mathcal{D}}(\boldsymbol{x}_*)\right),$$

where

$$\nabla \mu_{\mathcal{D}}(\boldsymbol{x}_*) = \nabla \mu(\boldsymbol{x}_*) + \nabla \kappa(\boldsymbol{x}_*)^\top C^{-1}(\boldsymbol{y} - \boldsymbol{m})$$
$$\nabla k_{\mathcal{D}}(\boldsymbol{x}_*) = \nabla k(\boldsymbol{x}_*, \boldsymbol{x}_*)\nabla^\top - \nabla \kappa(\boldsymbol{x}_*)^\top C^{-1} \nabla \kappa(\boldsymbol{x}_*).$$

This GP is the posterior distribution over the gradient based on zeroth-order information, meaning observation of the objective function. It is also possible to incorporate gradient observation if available. For a depiction of a one-dimensional GP and its derivative, see Figure 2. The posterior over derivatives is a vector-valued GP for $d > 1$.

3.2 Gradient uncertainty

In our method, the objective of an experiment is finding a descent direction by minimizing the gradient uncertainty for a given parameterization $\boldsymbol{x}_*$. We achieve this by an optimal design of experiments (DoE). We define an extended dataset $\mathcal{D}' = \mathcal{D} \cup \{(X', y')\}$ which includes future observations at X' and it's unknown value y'. We denote the total variance as the sum of the eigenvalues of the covariance matrix at $\boldsymbol{x}_*$, which is its trace $\mathrm{Tr}(\nabla k_{\mathcal{D}}(\boldsymbol{x}_*))$. The posterior total variance at $\boldsymbol{x}_*$ after b additional future observations at $X' \in \mathcal{X}^b$ is (cf. [9])

$$\alpha_{\mathrm{TV}}(x_*, X') = \mathrm{Tr}\left(\nabla k_{\mathcal{D}'}(\boldsymbol{x}_*)\right), \tag{2}$$

where $\nabla k_{\mathcal{D}'}(\boldsymbol{x}_*)$ is posterior variance based on the extended dataset $\mathcal{D}'$. The total variance in (2) does not depend on the unknown future observation y' and only on the location of this observation, allowing for an analytic expression of (2). Consequently, a DoE can be computed by minimizing (2) over the controller parameters to be evaluated in the next batch X_{DoE}

$$X_{\mathrm{DoE}} = \arg\min_{X' \in \mathcal{X} \times \mathbb{R}^{d \times b}} \alpha_{TV}(x_*, X'). \tag{3}$$

Minimizing the total variance is equivalent to minimizing the quadratic distance of samples from the gradient distribution, which also minimizes the worst-case gradient estimation error [11]. As an alternative to a DoE based on the total variance Nguyen et al. [10] propose to maximize the probability of descent.

4 Proposed method

The method proposed in this article is a local BO approach designed to deal with the crash-constrained optimization

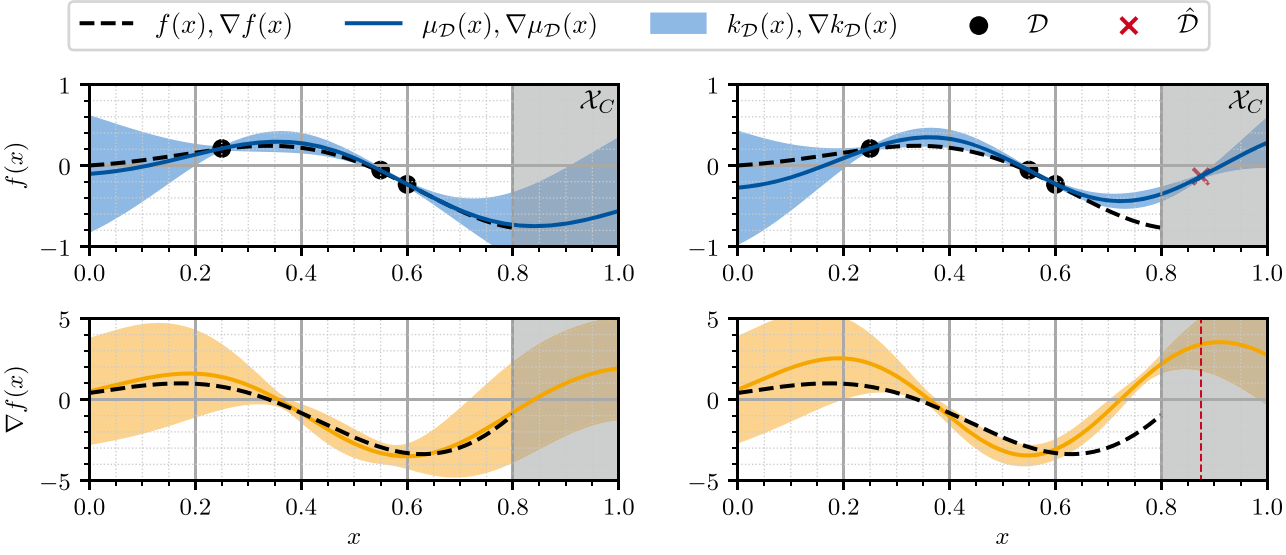


Figure 2: Left: A Gaussian process posterior (top) and its derivative (bottom). Right: The posterior with an additional virtual observation in the crash region $\mathcal{X}_c$. The crashed evaluation (red cross) cannot be evaluated, and a virtual observation is added instead. In this example, the virtual data point modifies the posterior such that the minimum of the posterior is not inside the infeasible region $\mathcal{X}_c$, and the gradient points away from it.

problems often encountered in controller tuning. It is an extension of the GIBO method proposed in Müller et al. [9] and incorporates virtual data points for crashed evaluations introduced by Stenger and Abel [4]. The resulting optimization algorithm is summarized in Algorithm 1.

4.1 Virtual data points for crashed evaluations

As stated in Section 1.1, the feasible domain $\mathcal{X}_S$ is unknown. The optimization algorithm might try to evaluate f outside of this region, crash, and not receive a value for y. Nevertheless, we must incorporate this failed evaluation in the GP model. A naive approach is to replace the function evaluation with a virtual observation of a fixed penalty for violating the crash constraint. Such penalties enable using a standard GP model over the unconstrained domain $\mathcal{X}$, allowing for arbitrary acquisition procedures including GIBO. However, a fixed penalty effectively corresponds to fitting an extended objective function with a large discontinuity at the boundary between $\mathcal{X}_S$ and $\mathcal{X}_C$. Discontinuities are difficult to model with GPs, especially since we assume the function is differentiable (cf. Assumption 1). Instead, virtual data points are adaptive penalties that are chosen such that their value is not 'too far' from the model predictions and, therefore, avoid large differences in function values while still guiding the optimization process away from observed crashes. Following Stenger and Abel [4], the adaptive penalty of the virtual observation is set to

$$\hat{y}_i = \max\left(\mu_D(\hat{\boldsymbol{x}}_i), \mu_D(\boldsymbol{x}_*)\right) + \beta\sqrt{k_D(\hat{\boldsymbol{x}}_i)}, \tag{4}$$

where the $\hat{\boldsymbol{x}}_i$ are the observed crash locations, $\boldsymbol{x}_*$ is the current parameterization, and $\beta > 0$ is a problem dependent parameter. The virtual observations are added to the dataset, resulting in an augmented dataset $\hat{\mathcal{D}} = \mathcal{D} \cup \{(\hat{X}, \hat{y})\}$. The parameter β determines the relative magnitude of the adaptive penalty with respect to the GP model. Essentially, it encodes how unexpected a crash is for the model. In general, determining a β for the problem at hand can be difficult. However, setting $\beta = 3$ empirically worked well. It means that the 'performance' of a crash is outside the 99 % confidence interval of the model. Furthermore, we enforce a lower bound so the penalty is always larger than the posterior mean of current parameterization $\boldsymbol{x}_*$. This ensures a non-negative slope between $\boldsymbol{x}_*$ and all crash locations $\hat{X}$, which can counteract cases where f exhibits steep gradients towards the constraint. An example of the effect of virtual observations on the GP posterior is shown in Figure 2.

4.2 Gradient information Bayesian optimization under crash constraints

We introduce a modification of the GIBO algorithm [9] for crash-constrained optimization problems called VDP-GIBO in Algorithm 1. The modifications to the original GIBO algorithm include (i) batch evaluations, (ii) a virtual dataset for the crashed observations, and (iii) resetting to a feasible parameterization if an update fails. The core idea of GIBO is to evaluate the parameters that reduce the uncertainty of the gradient at the current iterate x_* and then use the gradient estimate of the GP to perform gradient descent

$$\boldsymbol{x}_{*,k+1} = \boldsymbol{x}_{*,k} - \eta_k \nabla \mu_D(\boldsymbol{x}_k),$$

where k denotes the iterate of VDP-GIBO. Before each gradient step, the function is evaluated at b locations that minimize the total variance of $p(\nabla f(\boldsymbol{x}_*))$ (Section 3.2), where b is the batch size. Specifically, we implement (2) in an automatic differentiation library and use a standard optimizer to find the DoE.

Our proposed algorithm VDP-GIBO needs to deal with crashes during evaluation and updates. If an evaluation crashes, we utilize the virtual dataset introduced in Section 4.1 to build a posterior over the whole search domain $\mathcal{X}$. The virtual data points discourage further exploration near the crashed locations. Additionally, the adaptive penalty biases the gradient away from these areas, as illustrated in Figure 2. After each update, the algorithm evaluates the new parameterization to ensure it is feasible. If not, x_* is reset to a known feasible location from past evaluations X. Assuming deterministic crashes, resetting to a known feasible evaluation guarantees a viable solution after each improvement step. Specifically, we reset the parameters to

Algorithm 1: VDP-GIBO

Input: Initial parameter $\boldsymbol{x}_0$, batch sizes b_k, stepsizes η_k
$\boldsymbol{x}_* \leftarrow \boldsymbol{x}_0$, $k = 0$
$y \leftarrow f(\boldsymbol{x}_0) + \epsilon$, $\mathcal{D} \leftarrow \{(\boldsymbol{x}_0, y)\}$ ▷ Optionally evaluate $\boldsymbol{x}_0$
repeat
  $X_{\text{DoE}} = \arg\min_{X'} \alpha_{TV}(x_*, X')$ ▷ next batch (cf. (3))
  $\boldsymbol{y} \leftarrow f(X_{\text{DoE}}) + \epsilon$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(X_{\text{DoE}}, \boldsymbol{y})\}$ ▷ update dataset
  Build virtual dataset $\hat{\mathcal{D}}$ (cf. Section 4.1)
  $\boldsymbol{x}_* \leftarrow \boldsymbol{x}_* - \eta_k \nabla \mu_{\hat{D}}(\boldsymbol{x}_k)$ ▷ gradient update
  $y \leftarrow f(\boldsymbol{x}_*) + \epsilon$ ▷ evaluate x_* and update dataset
  if $\boldsymbol{x}_*$ is not feasible **then**
    $\boldsymbol{x}_* \leftarrow \arg\min_{x \in X} \mu_{\hat{D}}(x)$ ▷ reset to a feasible x
  end if
  Optimize GP hyperparameters.
until $\left|\hat{\mathcal{D}}\right| \geq K$
return $\boldsymbol{x}_*$

the past evaluation with the minimal posterior mean $\boldsymbol{x}_* = \arg\min_{x \in X} \mu_{\hat{D}}(x)$.

In general, GIBO can be used with any twice differentiable covariance function. However, we opt for the Gaussian kernel for our experiments since controller tuning objective functions are often smooth. The Gaussian kernel is defined as (cf. [30])

$$k_G(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f \exp\Big(\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^\top L^{-2}(\boldsymbol{x} - \boldsymbol{x}')\Big),$$

where σ_f is the signal variance and L is the positive-definite lengthscale matrix. Using this kernel allows us to automatically adapt the stepsize η_k of the gradient update to the lengthscale of the GP

$$\eta_k = \frac{\hat{\eta}_k}{\sqrt{\nabla \mu_D(\boldsymbol{x}_*)^\top L^{-2} \nabla \mu_D(\boldsymbol{x}_*)}}, \tag{5}$$

where $\hat{\eta}_k$ is a chosen step size. Equation (5) scales the stepsize based on the expected correlations of function values. See Müller et al. [9] for more details in gradient normalization with GP lengthscales.

5 Simulation results

This section provides an overview of the coupled tank system used to assess the performance of VDP-GIBO in controller tuning tasks. We also introduce the control algorithms under consideration and present the results of the simulated controller tuning process using VDP-GIBO.

5.1 Process description

The subject of examination in this study is a coupled tank system, illustrated in Figure 3. The system consists of three tanks, labeled B2, B3, B4 as well as one reservoir tank, B1. The pump P1, and the valves V_1 and V_6 can be actuated by the controller. The objective of the control is formulated in terms of the water levels in the three tanks, V_2, V_3 and V_4 respectively. We formulate a non-linear state space representation of the system as

$$\dot{\boldsymbol{s}} = \zeta(\boldsymbol{s}, \boldsymbol{a}), \tag{6}$$

where ζ is the dynamics, $\boldsymbol{s} = \begin{bmatrix} V_2 & V_3 & V_4 \end{bmatrix}^\top$ are the states and $\boldsymbol{a} = \begin{bmatrix} U_P & U_{V1} & U_{V6} \end{bmatrix}^\top$ are the inputs of the system. The coupled tank system used here is described in more detail in Scheurenberg et al. [31]. Despite careful modeling and empirical validation, there are discrepancies between the model and the dynamics. For the control algorithms we linearize the model at the operating point P with $\boldsymbol{s}_P = \begin{bmatrix} 8 & 6 & 5 \end{bmatrix} \times 10^{-3}$ and $\boldsymbol{a}_P = \begin{bmatrix} 70.7 & 43.3 & 44.7 \end{bmatrix}$ and discretize it with with 10 Hz (cf. [31]).

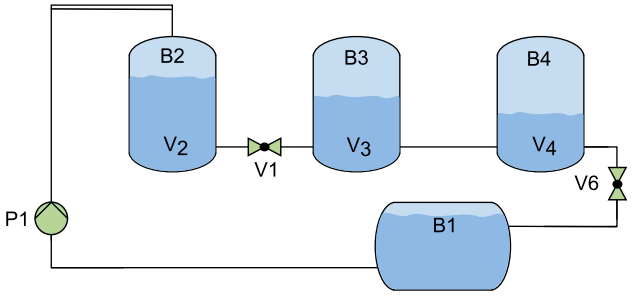


Figure 3: Diagram of the coupled tank system, with controllable pump and valves.

5.2 Process control

Three common controller configurations are employed to assess the automatic tuning algorithm VDP-GIBO: a cascaded control system using PI controllers, as well as an LQI controller and a linear MPC scheme. In the following, we will briefly overview the control loop structures. Additionally, we introduce the respective tuning parameters and objective functions of the five optimization problems summarized in Table 1.

The cascaded control comprises two distinct feedback loops. The outer loop regulates the water level V_4 by modifying the volumetric flow entering tank B2. The inner loop regulates this volumetric flow entering tank B2, by using U_P as the only controlled variable. The volumetric flow generated by the pump is accessible as a measured variable within this control system. Here, we consider two cases. In the first case, we tune only the outer loop: $\boldsymbol{x} = \begin{bmatrix} k_{\text{p,out}}, k_{\text{i,out}} \end{bmatrix}$. In the second case, all parameters for the cascaded structure are optimized. Furthermore, the position of valve V_1 is a tuning variable but is kept constant over one episode: $\boldsymbol{x} = \begin{bmatrix} k_{\text{p,out}}, k_{\text{i,out}}, k_{\text{p,in}}, k_{\text{i,in}}, U_{V1} \end{bmatrix}$.

Both the LQI and the MPC are multiple-input multiple-output (MIMO) control algorithms. The controlled variables

Table 1: Summary of the simulation test cases.

Case name	No. of params.	Objective	Crash V_2
PI ($V_2 < 8$ l)	2	RMSE	8 l
PI ($V_2 < 7$ l)	2	RMSE	7 l
Cascaded PI	5	RMSE	8 l
MPC + EKF	6	MAE	7.5 l
LQI	8	MAE	7.5 l

are the water levels V_2, V_3, V_4 and the manipulated variables are U_P, U_{V1}, U_{V6}. The MPC and the LQI both use a linear model obtained by linearizing (6) at a fixed operating point (cf. Section 5.1).

Since the process is non-linear, a linear MPC is not guaranteed to achieve zero steady-state error. An extended Kalman filter (EKF), used as a disturbance estimator, addresses this issue. The EKF and the MPC have several tuning parameters, such as the weighting matrices and the MPC control and prediction horizons. In principle, BO can also optimize the horizons [28], but here they are fixed to the values in Scheurenberg et al. [31]. For this study, we optimize the entries of the weighting matrix $\mathbf{Q}_{\text{MPC}} = \text{diag}[10^{x_1}, 10^{x_2}, 10^{x_3}]$ penalizing the tracking error. Additional tuning parameters are the entries corresponding to the disturbance process noise of the EKF $\mathbf{Q}_{\text{EKF}} = \text{diag}[1, 1, 1, 10^{x_4}, 10^{x_5}, 10^{x_6}]$, resulting in a total of 6 optimization variables. This parameterization enables tuning the weighting matrices over many orders of magnitudes and, empirically, yields well tuned controllers [28].

Similarly to an MPC, a standard LQR is not guaranteed to achieve zero steady-state error for a non-linear process. Therefore, using LQI adds integral error states to the state vector. For the LQI, all diagonal entries of the weighting matrices are optimized: $\mathbf{R}_{\text{LQI}} = \text{diag}[1, 10^{x_1}, 10^{x_2}]$, $\mathbf{Q}_{\text{LQI}} = \text{diag}[10^{x_3}, \dots, 10^{x_8}]$.

In both PI-tuning cases, one episode consists of one step of the reference for V_4. The objective function is the root mean squared tracking error (RMSE)

$$f_{\text{RMSE}}(\boldsymbol{x}) = \mathbb{E}\left[\sqrt{\frac{1}{T}\int_0^T (V_4(t, \boldsymbol{x}) - V_{4,\text{ref}}(t))^2 \mathrm{d}t}\right],$$

where T is the episode length. In cases where the controller is not used for a short episode but for continuous operation, a representative episode with typical disturbances must be defined by domain experts. This is a general problem for all data-based tuning methods. A typical scenario is the closed-loop step response. Here, the objective is evaluated using a single experiment: a noisy approximation of the true expectation. In the MPC and LQI cases, the objective function and the reference trajectory are chosen differently to highlight the broad applicability of BO. Here, the weighted sum of the mean absolute tracking error (MAE) of each tank is minimized

$$f_{\text{MAE}}(\boldsymbol{x}) = 0.5 f_{V,2}(\boldsymbol{x}) + 0.25 f_{V,3}(\boldsymbol{x}) + 0.25 f_{V,4}(\boldsymbol{x}),$$

where

$$f_{V,i}(\boldsymbol{x}) = \mathbb{E}\left[\frac{1}{T}\int_0^T |V_i(t, \boldsymbol{x}) - V_{i,\text{ref}}(t)| \mathrm{d}t\right].$$

Instead of one reference step, two consecutive reference steps for all water volumes are evaluated.

A simulation is aborted, i.e., a crash occurs, in case the first tank exceeds a critical volume. In practice, this prevents a tank from overflowing. For the PI-case, this crash V_2 is set to 7 l and 8 l, respectively. In the cascaded PI case, 8 l and for the remaining cases, 7.5 l is chosen.

5.3 Tuning results

We evaluate the performance of VDP-GIBO in the five settings described in the previous section. Each control algorithm is tuned ten times with randomized initial controller parameterization and different noise realizations. The initial controller parameterization is chosen from a small set in the feasible region $\mathcal{X}_S$ where the performance is relatively low. We use the same hyperparameters in all experiments (Table 2). These parameters were obtained manually from initial experimentation with the PI controller. Since the hyperparameters were not tuned to the specific problems and were chosen based on basic knowledge of the problem domain, we conclude that the controller tuning with VDP-GIBO is not very sensitive to the choice of hyperparameters.

We compare the tuning result of VDP-GIBO with the result of random search, where we draw controller parameterization uniformly from the search domain $\mathcal{X}$, evaluate them and choose the best one. We use the same number of evaluations for the random search baseline as for VDP-GIBO. The optimization results are shown in Figure 4. The PI tuning problems (Figure 4, top) are relatively easy and random search can solve them within a few evaluations. Our method recovers similar solutions within the given budget and even finds slightly better solutions for the more constrained problem (Figure 4, top-middle). However, for these easy tuning problems, VDP-GIBO usually takes more evaluations than random search. The reason is that the

Table 2: VDP-GIBO hyperparameters for all simulations and experiments.

Hyperparamter	Value
b_k	$d + 1$
$\hat{\eta}_k$	0.25 … 0.125 (with cosine decay)
L	$0.25\, I$
σ_f	0.5
$\mu(x)$	1
β	3

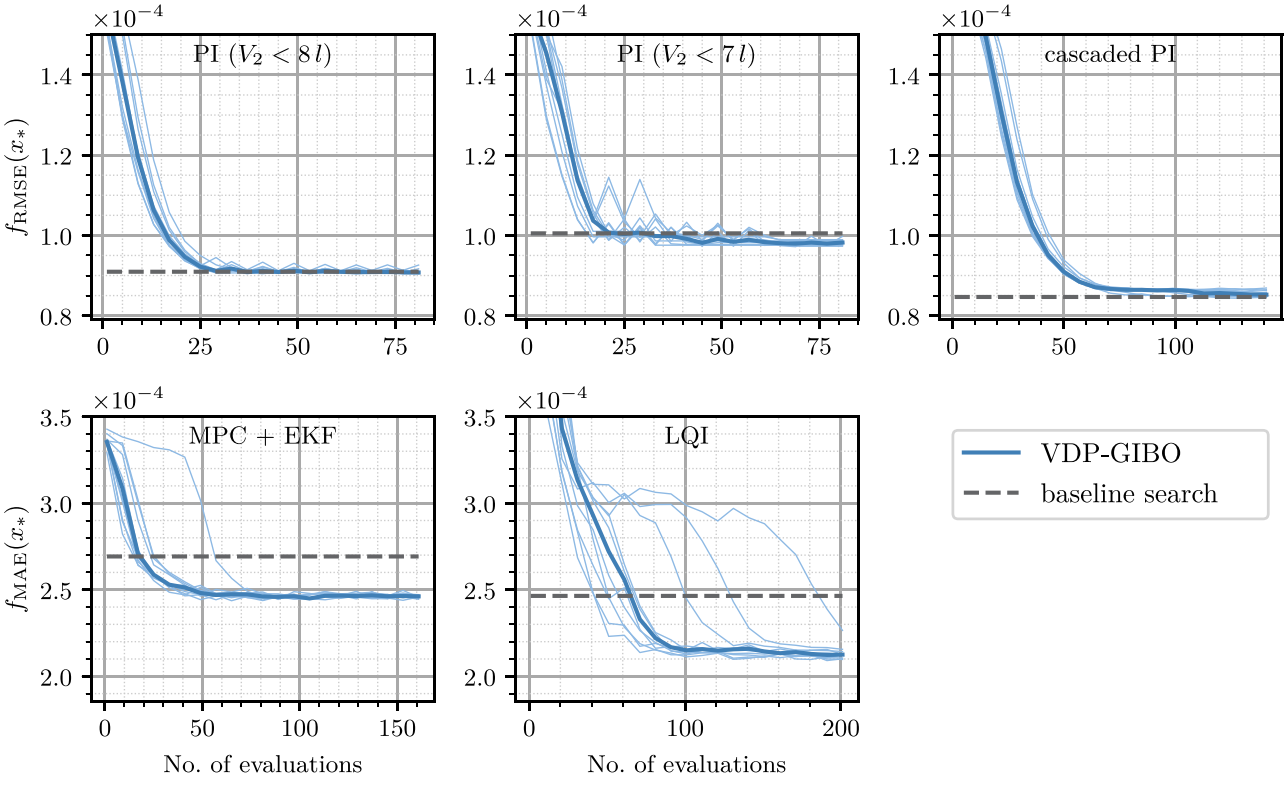


Figure 4: Simulation results on crash-constrained controller tuning problems: VDP-GIBO is able to solve 2-(PI), 4-(cascaded PI), 6-(MPC) and 8-dimensional (LQI) controller tuning problems in a handful of evaluations. The controller performance shown as the median over 10 runs with randomized initial controller parameters. The individual runs are shown as thin lines and demonstrate the low variability in tuning results with the proposed method. As baseline (dashed line), we draw parameters uniformly at random from the search domain and chose the best evaluation. The number of evaluation is the same as for VDP-GIBO. Please note that the objective functions are different between the PI and the MIMO (LQI and MPC) controllers.

initial guess is purposefully poor, and VDP-GIBO performs a local search, requiring a few steps before leaving this high-cost initial region. For the more complex and higher dimensional tuning problems (Figure 4, bottom), random search is not a viable tuning strategy, and VDP-GIBO finds controllers with significantly better performance within the same evaluation budget, highlighting the data-efficiency of our proposed method. Since the difficulty of a problem is unknown *a priori*, VDP-GIBO yields more consistent results.

The resulting control performances are consistent with control engineering intuition: Tightening the upper bound for V_2 results in a worse tracking behavior. Additionally, adding degrees of freedom to the tuning task by adjusting the inner control loop and the first valve position can increase closed-loop performance. For the LQI tuning problem, we allow for the relative weighting between the different penalties, R_{LQI}, on the control input. In contrast, for the MPC, the weighting matrix for the input is fixed. Generally, control engineering expertise is required to choose influential tuning parameters for a given controller structure. Additionally, it emphasizes the importance of VDP-GIBO being able to address high-dimensional tuning problems.

For the two-dimensional tuning problems, we show the evaluations of VDP-GIBO in the parameter space in Figure 5. The algorithm is able to find a local optimum without exploring the whole search space $\mathcal{X}$, leading to a data-efficient tuning process and relatively few crashes, even when a large part of the space is infeasible. For the problem with tighter constraints on V_2 a larger part of the search space, including the unconstrained optimum, is infeasible. The virtual data points enable a GP model over the whole search space, and VDP-GIBO converges to a feasible estimate.

6 Experimental results

We repeated the tuning experiments for the PI and the cascaded PI controller on a hardware test bed that implements the process described in Section 5.1. The testbed is part of the model factory at the *Institute of Automatic Control, RWTH*

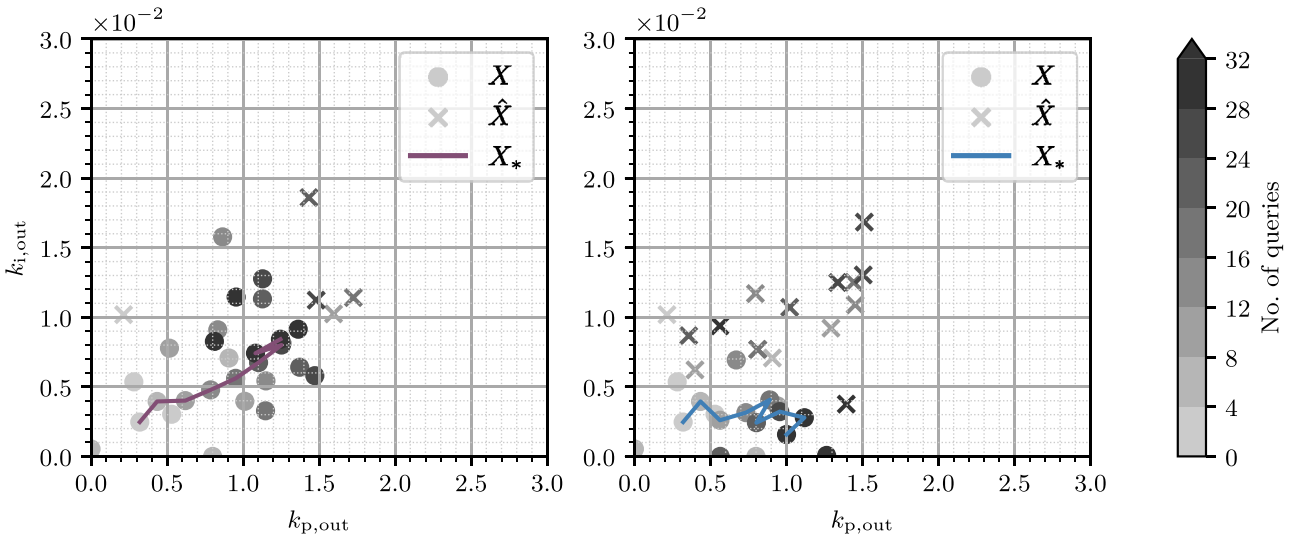


Figure 5: Evaluations in the parameter space for PI ($V_2 < 8$ l) (left) and PI ($V_2 < 7$ l) (right). We show the first eight improvement steps and the corresponding evaluation locations X, improvement steps X_* and crashes $\hat{X}$. Due to the tighter constraints on the right, the feasible optimum changes. The virtual observations change the gradient such that the algorithm can estimate this new local optimum. The majority of the parameter space remains unexplored, increasing data-efficiency.

Figure 6: Experimental setup: The top three tanks correspond to tanks B2, B3, and B4 (from left to right). The black boxes in the middle of the image are the pneumatic valves V_1 and V_6 (from left to right). At the bottom of the picture are the pump and the reservoir tank B1.

Aachen University and is depicted in Figure 6. Analogous to the description of the simulation model, water is pumped into tank B2 by pump P1. The volumetric flow generated by the pump as a function of the control input is measured using a flow sensor. The pneumatic valve V_1 controls the volume flow from B2 to B3. Subsequently, tank B3 is connected to tank B4 without a controllable valve. Tank B4 is connected to the reservoir tank B1 via V_6. Each tank has a level sensor. All hyperparameters of VDP-GIBO are the same as in Section 5.3.

For the first two experiments, we tuned the outer control loop of the cascaded PI controller: $\boldsymbol{x} = \left[k_{p,out}, k_{i,out}\right]$. The inner controller was set to hand-tuned values. In the first experiment, we set the crash constraint on the maximum water level in all tanks to 7.5 l. When the water level reaches this level, the experiment is aborted. In this setting, most controller parameterizations are feasible, and due to the local exploration of VDP-GIBO, all evaluations during tuning were feasible. We ran the experiment for eight iterations, corresponding to 33 evaluations or approximately 1.5 h. The controller performance improved by ca. 33 % (Figure 7). Most of the improvement was achieved in the first three gradient steps, highlighting the efficiency of local BO for controller tuning.

The second experiment is run with a maximum water level of 6.1 l. With this constraint, many parameters lead to an emergency system stop, preventing the experiment from completion. In the tuning experiment, three controller

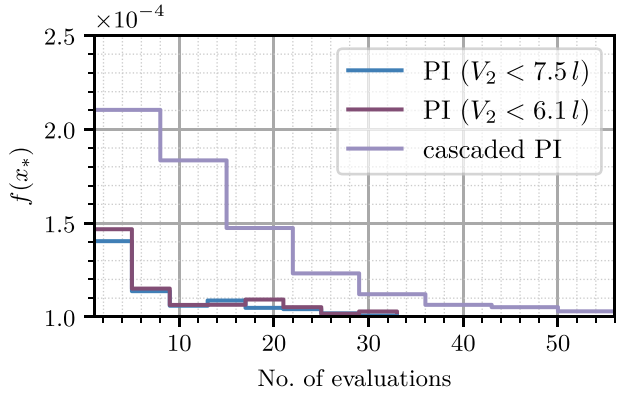


Figure 7: Experimental results for crash-constrained PI control: control performance of the PI controller during the tuning process. Despite the crash constraints, VDP-GIBO is able to use the data efficiently and improve by approx. 33 % for the PI controller and by 50 % for the cascaded PI controller.

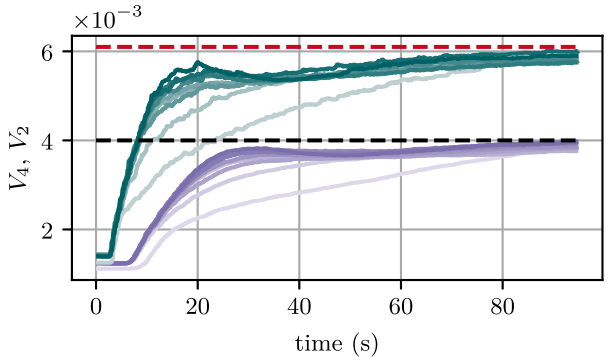


Figure 8: Time domain of the PI controller: step response of V_4 (purple) and V_2 (petrol) with the PI controller. After tuning, the state tracks the desired reference (black) significantly better. Darker colors indicate the behavior later in the tuning process. The state V_2 stays below the constraint (red).

evaluations crashed, with two crashes during the exploration phase and one during an update. Our proposed algorithm VDP-GIBO is still able to achieve a similar tuning result as in the previous setting despite the more difficult tuning task (Figure 7). The time domain behavior of the controlled and constrained state is depicted in Figure 8.

In the third experiment, we jointly tune the cascaded PI controller and the position of valve V_1: $\boldsymbol{x} = \left[k_{\text{p,out}}, k_{\text{i,out}}, k_{\text{p,in}}, k_{\text{i,in}}, U_{V1}\right]$. The initial parameters are chosen such that the initial performance is poor. Due to the higher dimensional search space, eight iterations of VDP-GIBO require a budget of 54 evaluations or 2.7 h. The algorithm improves the control performance by approximately 50 % and, due to its local exploration behavior, never leaves the feasible region.

7 Conclusions

In this article, we propose VDP-GIBO, a novel controller tuning algorithm for optimization problems under crash constraints, and demonstrate its data-efficiency on three standard control algorithms: PI control, LQI, and MPC. While the proposed algorithm has a set of hyperparameters, these were always the same for all the results presented in the paper. This points towards the applicability of the algorithm for general controller tuning problems. However, other tuning problems may require additional effort and evaluations to find suitable parameters.

Sample-efficient and intuitive controller tuning, especially for well-proven policy/controller structures such as PI, LQI, and MPC using VDP-GIBO, can lead to overall higher control performance by combining control engineering expertise with data-driven techniques. Control engineering expertise is required to choose a suitable controller structure and formulate the tuning task by defining the objective function and tuning parameters. The data-driven optimization explores the search space locally, resulting in – compared to global exploration – gradual changes in time domain behavior. This enables a more intuitive understanding of the tuning progress.

Automated tuning can help control engineers to fairly compare the applicability of different controller structures to practical control engineering tasks. Tuning structures with respect to identical objective functions eliminate the influence of hand-picked parameter values on the comparison. The applicability of VDP-GIBO to higher dimensional problems enables the control engineer to design efficient controller structures with many parameters.

This paper assumes that a mechanism to detect a crash and reset the system to its initial state is present. Such resets might be challenging for fully automated controller tuning and require a backup controller to take over in case of failures (e.g., [32], [33]).

Acknowledgments: The authors thank P. Brunzema, J. Menn, and F. Solowjow for constructive criticism of the manuscript.
Research ethics: Not applicable.
Author contributions: The authors have accepted responsibility for the entire content of this manuscript and approved its submission.
Competing interests: The authors state no competing interests.
Research funding: None declared.

Data availability: The raw data can be obtained on request from the corresponding author.

References

[1] M. Jelali, "An overview of control performance assessment technology and industrial applications," *Control Eng. Pract.*, vol. 14, no. 5, pp. 441—466, 2006.

[2] 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 Trans. Robot.*, vol. 36, no. 2, pp. 328—347, 2020.

[3] J. A. Paulson, F. Sorourifar, and A. Mesbah, "A tutorial on derivative-free policy learning methods for interpretable controller representations," in *2023 American Control Conference (ACC)*, 2023, pp. 1295—1306.

[4] D. Stenger and D. Abel, "Benchmark of bayesian optimization and metaheuristics for control engineering tuning problems with crash constraints," 2022, *arXiv preprint arXiv:2211.02571*.

[5] F. Bachoc, C. Helbert, and V. Picheny, "Gaussian process optimization with failures: classification and convergence proof," *J. Global Optim.*, vol. 78, no. 3, pp. 483—506, 2020.

[6] 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)*, 2016, pp. 270—277.

[7] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Ann. Math. Artif. Intell.*, vol. 76, nos. 1—2, pp. 5—23, 2016.

[8] A. Marco, D. Baumann, M. Khadiv, P. Hennig, L. Righetti, and S. Trimpe, "Robot learning with crash constraints," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1439—1446, 2021.

[9] S. Müller, A. von Rohr, and S. Trimpe, "Local policy search with Bayesian optimization," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 20708—20720, 2021.

[10] Q. Nguyen, K. Wu, J. Gardner, and R. Garnett, "Local Bayesian optimization via maximizing probability of descent," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 13190—13202, 2022.

[11] K. Wu, K. Kim, R. Garnett, and J. R. Gardner, "The behavior and convergence of local bayesian optimization," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[12] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *International Conference on Machine Learning*, 2010, pp. 1015—1022.

[13] O. Andersson, M. Wzorek, P. Rudol, and P. Doherty, "Model-predictive control with stochastic collision avoidance using Bayesian policy optimization," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4597—4604.

[14] H. Chen, S. Bowels, B. Zhang, and T. Fuhlbrigge, "Controller parameter optimization for complex industrial system with uncertainties," *Meas. Control.*, vol. 52, nos. 7—8, pp. 888—895, 2019.

[15] Z. Chen, C. Heckman, S. Julier, and N. Ahmed, "Weak in the NEES? Auto-tuning kalman filters with bayesian optimization," in *2018 21st International Conference on Information Fusion (FUSION)*, 2018, pp. 1072—1079.

[16] J. Marzat, H. Piet-Lahanier, and E. Walter, "Min-max hyperparameter tuning, with application to fault detection," *IFAC Proc. Vol.*, vol. 44, no. 1, pp. 12904—12909, 2011.

[17] M. Khosravi, V. Behrunani, R. S. Smith, A. Rupenyan, and J. Lygeros, "Cascade control: data-driven tuning approach based on Bayesian optimization," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 382—387, 2020.

[18] D. Stenger, M. Nitsch, and D. Abel, "Joint constrained Bayesian optimization of planning, guidance, control, and state estimation of an autonomous underwater vehicle," in *2022 European Control Conference (ECC)*, 2022, pp. 1982—1987.

[19] 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)*, 2019, pp. 5850—5856.

[20] M. Khosravi, C. König, M. Maier, R. S. Smith, J. Lygeros, and A. Rupenyan, "Safety-aware cascade controller tuning using constrained Bayesian optimization," *IEEE Trans. Ind. Electron.*, vol. 70, no. 2, pp. 2128—2138, 2023.

[21] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics," *Mach. Learn.*, vol. 112, no. 10, pp. 3713—3747, 2021.

[22] D. Stenger, R. Ritschel, F. Krabbes, R. Voßwinkel, and H. Richter, "What is the best way to optimally parameterize the MPC cost function for vehicle guidance?" *Mathematics*, vol. 11, no. 2, p. 465, 2023.

[23] G. Makrygiorgos, A. D. Bonzanini, V. Miller, and A. Mesbah, "Performance-oriented model learning for control via multi-objective Bayesian optimization," *Comput. Chem. Eng.*, vol. 162, p. 107770, 2022.

[24] L. Fröhlich, E. Klenske, J. Vinogradska, C. Daniel, and M. Zeilinger, "Noisy-input entropy search for efficient robust Bayesian optimization," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, Volume 108 of Proceedings of Machine Learning Research*, S. Chiappa and R. Calandra, Eds., 2020, pp. 2262—2272.

[25] J. A. Paulson, G. Makrygiorgos, and A. Mesbah, "Adversarially robust Bayesian optimization for efficient auto-tuning of generic control structures under uncertainty," *AIChE J.*, vol. 68, no. 6, 2022, Art. no. e17591.

[26] M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: preference-based global optimization under unknown constraints with applications to controller calibration," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 5, pp. 1—12, 2021.

[27] 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.

[28] D. Stenger, M. Ay, and D. Abel, "Robust parametrization of a model predictive controller for a CNC machining center using Bayesian optimization," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 10388—10394, 2020.

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

[30] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Cambridge, MIT Press, 2006.

[31] 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.
[32] D. Baumann, A. Marco, M. Turchetta, and S. Trimpe, "GoSafe: globally optimal safe robot learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4452—4458.
[33] B. Schürmann, M. Klischat, N. Kochdumper, and M. Althoff, "Formal safety net control using backward reachability analysis," *IEEE Trans. Autom. Control*, vol. 67, no. 11, pp. 5698—5713, 2022.

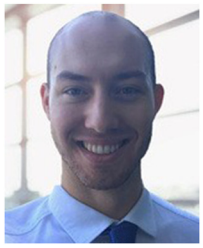
Bionotes

Alexander von Rohr
Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Aachen, Germany;
vonrohr@dsme.rwth-aachen.de
https://orcid.org/0000-0002-0005-0310

Alexander von Rohr is a PhD student at the Institute for Data Science in Mechanical Engineering, RWTH Aachen University. Previously, he earned a M.Sc. in computer science from the University of Lübeck and a B.Eng. in electrical engineering from BHT Berlin. He is an associated student of the International Max Planck Research School for Intelligent Systems (IMPRS-IS). His main research interests are Bayesian optimization and learning-based control.

David Stenger
Institute of Automatic Control, RWTH Aachen University, Aachen, Germany;
d.stenger@irt.rwth-aachen.de
https://orcid.org/0000-0003-3747-4499

David Stenger is a postdoctoral researcher at RWTH Aachen University. He earned a M.Sc. and B.Sc. in Mechanical and Process Engineering from the Technical University of Darmstadt and received his Ph.D. degree (Dr.-Ing.) from RWTH Aachen University in 2023. His main research interests are in Bayesian optimization and control engineering.



Dominik Scheurenberg
Institute of Automatic Control, RWTH Aachen University, Aachen, Germany;
d.scheurenberg@irt.rwth-aachen.de
https://orcid.org/0000-0002-1044-5631

Dominik Scheurenberg is a PhD student at RWTH Aachen University. Before his PhD, he received an M.Sc. in Automation Engineering, a B.Sc. in Electrical Engineering and Information Technology both from RWTH Aachen University and a B.Sc. in Biomimetics from Westphalian University of Applied Sciences. His main research interests are in model predictive control and machine learning.

Sebastian Trimpe
Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Aachen, Germany;
trimpe@dsme.rwth-aachen.de
https://orcid.org/0000-0002-2785-2487

Sebastian Trimpe is professor and head of the Institute for Data Science in Mechanical Engineering at RWTH Aachen University. Since starting at RWTH in 2020, he has been also a director at the RWTH Center for Artificial Intelligence, and became its co-chair in 2023. Before joining RWTH, he was an independent Research Group Leader at the Max Planck Institute for Intelligent Systems in Stuttgart and Tübingen. Sebastian received his Ph.D. degree (Dr. sc.) from ETH Zurich in 2013. His main research interests are in control theory and design, machine learning, networked systems, and robotics.