Control design via Bayesian Optimization with safety constraints

Xavier Bombois, Marco Forgione

Control design via Bayesian Optimization with safety constraints

Xavier Bombois[1] and Marco Forgione[2]

***Abstract*— Bayesian Optimization is a powerful machine-learning tool enabling automated design of fixed-structure controllers. A sequence of closed-loop calibration experiments is performed, and the next configuration to be tested is selected by the optimization algorithm in order to minimize an objective function measured directly on the real system. While the approach has been shown to be effective, its applicability is limited in certain domains by safety considerations, as the algorithm may suggest controller configurations which lead to dangerous behaviours in some of the calibration experiments. In this paper, we modify the standard Bayesian Optimization algorithm by introducing explicit constraints for safe exploration of the controller configuration space. The constraints are derived based on a preliminary model of the process dynamics, which is assumed to be available. Aspects for efficient implementation of the proposed methodology are discussed. Simulation examples highlight the advantage of the proposed methodology for controller calibration over the plain Bayesian Optimization algorithm.**

I. Introduction

In recent years, data-driven controller calibration methodologies based on global derivative-free optimization algorithms have been shown to be tremendously effective [1]. To cite a few examples, optimal selection of position controllers for quadrotor drones is shown in [2], while automatic tuning of PID gains (along with feedback-linearization parameters) for robotic manipulators is demonstrated in [3]. An automotive application to the design of active disturbance rejection controllers for throttle valve regulation is described in [4].

To apply these data-driven calibration methodologies, the user must specify: (i) a control objective which can be evaluated in terms of *measured* input/output data and (ii) a search space where the calibration parameters are allowed to vary. The optimization algorithm is then executed in an *interactive* manner, where function evaluations correspond to closed-loop experiments to be performed on the physical set-up. This is in sharp contrast with standard numerical optimization, where a simulated objective function is repeatedly evaluated.

The need to perform real experiments as part of the optimization loop entails specific challenges. To be of practical interest, the algorithm should be *sample-efficient*, namely it should be able to propose good candidate controllers (although not necessarily *optimal*) within a limited number of iterations. Furthermore, it should be tolerant to the presence of noise in the objective function, as the latter is affected by all sources of experimental variability. Finally, all the configurations suggested by the algorithm should be *safe*, in the sense that they should not lead to catastrophic failures and/or damage of the experimental set-up.

For its sample-efficiency and noise-tolerance properties, the Bayesian Optimization (BO) algorithms [5] is a common choice in this context. In a nutshell, BO uses all the available function evaluations to construct a probabilistic *surrogate* description of the objective function. This probabilistic surrogate is then used to choose the next point to be tested according to a trade-off criterion which balances exploitation (selection of a point close to the optimum of the surrogate objective) and exploration (selection of a point with high uncertainty). The explicit exploration goal is the distinguishing feature of BO with respect to traditional surrogate function optimization methods [6]. On the one hand, by exploring the whole search space systematically, the algorithm is guaranteed not to get stuck in configurations which are only locally optimal. On the other hand, all controller configurations—including the ones that pose safety concerns—may eventually be suggested by the algorithm for testing.

In this paper, we propose a modification of the standard BO algorithm to ensure safe exploration of the controller configuration space. Our approach is based on a model of the underlying system, which is used to restrict the search space for BO to safe configurations for experimentation. In particular, we exclude from the search space those configuration whose performance is below a certain threshold when applied (in simulation) to the model. The rationale of this approach is that a controller leading to poor performance on the model (or even de-stabilizing this model) should not be tested, as they are also likely to perform poorly on the real system.

Safe exploration for BO is an active area of investigation in the general Machine Learning community. For instance, the SafeOpt algorithm [7] introduces a safe exploration set based on *a priori* regularity assumptions (i.e. bounded norm, Lipschitz-continuity) on the unknown objective function. At each iteration, it selects a new point according to the competing goals of expanding the safe set (by sampling points close to its boundary) and optimizing the objective function (by sampling points close to the optimum of the surrogate). The SafeOpt-MC algorithm [8] extends SafeOpt by including multiple constraint functions, all assumed to be sufficiently regular. Finally, GoOSE [9] aims at improving the sample-

*The activities of Marco Forgione have been supported by HASLER STIFTUNG under the project DEALING: DEep learning for dynamicAL systems and dynamical systems for deep learnING.

[1]Xavier Bombois is with Laboratoire Ampère, Ecole Centrale de Lyon, Université de Lyon, 36 avenue Guy de Collongue, Ecully, France and with Centre National de la Recherche Scientifique (CNRS), France xavier.bombois@ec-lyon.fr

[2]Marco Forgione is with IDSIA Dalle Molle Institute for Artificial Intelligence USI-SUPSI, Via la Santa 1, Lugano-Viganello CH-6962, Switzerland marco.forgione@supsi.ch

efficiency of the above methodologies, by expanding the safe exploration region only in the directions where the objective function is expected to decrease.

Compared to those methodologies, we do not rely on regularity assumptions on the objective function and instead exploit domain knowledge to define safe regions for controller parameter selection.

The rest of this paper is organized as follows. The overall control design framework is introduced in Section II and the proposed Bayesian Optimization method with safe parameter exploration is presented in Section III. In Section IV, examples showcasing the advantage of the proposed methodology over plain BO are illustrated. Conclusions and directions for further investigations are discussed in Section V.

II. Framework

Let us consider an uncertain continuous-time dynamical system $\mathcal{S}$ with a control input u and an output y. Our objective is to tune the parameters $\theta \in \Theta$, $\Theta \subseteq \mathbb{R}^{n_\theta}$ of a controller $K(\theta)$ in such a way that a given closed-loop performance objective is met (the search space Θ will be discussed in more details in the next section). The objective is quantified in terms of a given *cost function* $V(\theta)$ that we wish to minimize. This cost function $V(\theta)$ can e.g., be the energy of the difference between the desired output $y_{des}(t)$ and the output $y(t,\theta)$ of the system $\mathcal{S}$ when $K(\theta)$ is chosen as controller. In any case, we suppose that, for any value of $\theta \in \Theta$, the cost function $V(\theta)$ can be measured via an experiment on the closed-loop system $[K(\theta)\ \mathcal{S}]$ made up of the system $\mathcal{S}$ and the controller $K(\theta)$ i.e., the cost function $V(\theta)$ can be estimated/measured based of the input and output samples generated during this experiment. We will denote this measure by $\tilde{V}(\theta)$.

As an example, if $V(\theta)$ is chosen as proposed above (i.e., the energy of $y(t,\theta) - y_{des}(t)$), we can perform an experiment of sufficient duration NT_s (T_s is the sampling rate) on the loop $[K(\theta)\ \mathcal{S}]$ yielding a sampled version $y_n(\theta)$ ($n = 1, ..., N$) of the continuous-time output $y(t,\theta)$ at time instant $t = nT_s$ and we can then choose $\tilde{V}(\theta) = T_s \sum_{n=1}^{N}(y_n(\theta) - y_{des,n})^2$ where $y_{des,n}$ is the sampled version of the desired output $y_{des}(t)$ at time instant $t = nT_s$.

In our data-driven approach, the controller design task is seen as a global optimization problem, to be solved in a number n_{it} of iterations. Each iteration is characterized by an experiment on the system $\mathcal{S}$ with a different controller $K(\theta_k)$ (θ_k is the value of θ that is tested at Iteration k). After this experiment, we obtain the measurement $\tilde{V}(\theta_k)$ of $V(\theta_k)$ and we determine a non-parametric probabilistic model[1] (usually a Gaussian process model [10]) of the cost function $V(\theta)$ based on the measurements collected up to Iteration k. This probabilistic model (together with its uncertainty) is then used to suggest the next controller configuration θ_{k+1} to be tested in the next experiment, aiming at minimization of the cost $V(\theta)$.

[1]A closed-form expression of the cost V as a function of the design parameter vector θ is indeed not available due to the fact that the system $\mathcal{S}$ is unknown/uncertain.

The global-optimization problem is generally tackled via a so-called Bayesian Optimization (BO) algorithm [5]. As mentioned above, this algorithm will entail n_{it} experiments on the system $\mathcal{S}$ with different controllers $K(\theta)$. In these multiple experiments, certainly in the initial phase where the probabilistic model of $V(\theta)$ is very uncertain, some of these experiments could pertain to controllers that destabilize the system or give a performance that is so bad that it could harm the system. In this paper, we will provide a modified Bayesian Optimization algorithm that excludes such experiments using available prior knowledge. This prior knowledge will here take the form of a (linear) model M of the (possibly non-linear) system $\mathcal{S}$. As we will see in the sequel, the model M will be exploited to define a region of safe exploration. The rationale of this approach is that a controller leading to poor performance on the model M (or even de-stabilizing this model) should not be tested, as they are also likely to perform poorly on the real system $\mathcal{S}$.

III. Method

In this section, the proposed algorithm for control design with Bayesian Optimization is presented. First, the standard BO algorithm is described in Section III-A. Then, in Section III-B, the algorithm is modified to ensure a safe exploration.

As mentioned in the previous section, the BO algorithm aims at approaching the solution $\theta^\star$ of the following (generally non-convex) optimization problem:

$$\theta^\star = \arg\min_{\theta\in\Theta} V(\theta), \tag{1}$$

where Θ represents the search space for the controller parameters θ, this search space is generally defined via intervals in which each entry of θ may vary. It is important to stress that Θ is generally chosen as a large set and that it may therefore contain many parameter vectors θ leading to (very) poor performance $V(\theta)$.

A. Bayesian Optimization for controller tuning

Probabilistic model of $V(\theta)$: The key idea behind BO is to describe the (uncertain) relationship between the parameter vector θ and the corresponding cost function V by means of a surrogate *probabilistic* model of $V(\theta)$, and to iteratively update this model through Bayesian inference when new experiments are performed.

The most common probabilistic model used in BO is the *Gaussian Process* (GP) [10]. Let us analyze how this framework can be used for our BO algorithm and let us suppose that we are after Iteration k in this BO algorithm. After Iteration k, we have performed k experiments on $\mathcal{S}$ with k different controllers parametrized with different parameter values θ. Let us gather these k values in the set $\mathcal{T}_k = \{\theta_1, \theta_2, ..., \theta_k\}$ where θ_i ($i = 1, ..., k$) is the parameter value tested at Iteration i. These k experiments have led to k observations of the cost function. We also gather these observations in the set $\mathcal{T}_k^{\tilde{V}} = \{\tilde{V}(\theta_1), \tilde{V}(\theta_2), ..., \tilde{V}(\theta_k)\}$. Like in every Bayesian estimation process, we have to

suppose a prior distribution for the probabilistic model of $V(\theta)$ materializing the a-priori information on $V(\theta)$. In the GP framework, this prior distribution is under the form of a Gaussian distribution whose covariance matrix imposes a higher degree of similitude for values of $V(\theta)$ with θ that are close to each other in the parameter space. The GP framework also supposes a distribution for the *measurement error* $\tilde{V}(\theta) - V(\theta)$ (this difference is often supposed to be i.i.d. and to be Gaussian-distributed). This prior information is optimized[2] at each iteration based on the collected data $\mathcal{T}_k$ and $\mathcal{T}_k^{\tilde{V}}$. Then, the posterior distribution of the model of $V(\theta)$ given the data $\mathcal{T}_k$ and $\mathcal{T}_k^{\tilde{V}}$ can be easily computed for each value of $\theta \in \Theta$ due to the Gaussian prior assumption. This posterior distribution is a Gaussian distribution with mean $\mu_k(\theta)$ and variance $\sigma_k^2(\theta)$. At Iteration k, the probabilistic model $\mathcal{V}_k(\theta)$ of $V(\theta)$ at an arbitrary value $\theta \in \Theta$ is thus $\mathcal{N}(\mu_k(\theta), \sigma_k^2(\theta))$. The quantity $\mu_k(\theta)$ is therefore the best estimate of $V(\theta)$ given the observed data $\mathcal{T}_k$ and $\mathcal{T}_k^{\tilde{V}}$ (and the prior information) and $\sigma_k^2(\theta)$ determines the uncertainty of this estimate.

After each iteration, we can update the solution of our BO algorithm using the current probabilistic model of $V(\theta)$. After Iteration k, the current solution $\theta_{opt,k}$ is the element of $\mathcal{T}_k$ leading to the smallest value of $\mu_k(\theta)$ i.e.,

$$\theta_{opt,k} = \arg\min_{\theta \in \mathcal{T}_k} \mu_k(\theta). \tag{2}$$

This means that the (current) solution will always be a value of θ that has been tested on the system $\mathcal{S}$.

Acquisition function: The probabilistic model $\mathcal{V}_k(\theta)$ will also be used to determine the next value of θ that will be tested on the system $\mathcal{S}$. In Bayesian Optimization, the model's uncertainty information is explicitly taken into account to propose the most promising point to be tested in the next iteration. The choice is done according to a trade-off balancing exploration (namely, selecting a point where the expected value $\mu_k(\theta)$ of the cost is low) and exploration (namely, selecting a point where the variance $\sigma_k^2(\theta)$ of the model is high). This trade-off criterion is formalized in terms of an *acquisition function* $\mathcal{A}_k(\theta)$ such as the *expected improvement*

$$\mathcal{A}_k(\theta) = \mathbf{EI}(\theta) = \mathbb{E}\left[\max\{0, \mu_k(\theta_{opt,k}) - \mathcal{V}_k(\theta)\}\right], \tag{3}$$

which is the expected value of the improvement of the objective function with respect to the best previously tested points.

The Bayesian Optimization algorithm for controller tuning is described in Algorithm 1. Note that, instead of just considering one single parameter vector θ_1, Step 1 can also consider a small amount of parameter vectors randomly chosen in Θ in order to better initialize the optimization.

[2]The parameters describing this prior information (the so-called hyperparameters) are re-estimated at each iteration to make the observed data the most likely, by maximizing the *marginal* data likelihood, see [10].

Algorithm 1 Control design via Bayesian Optimization

1. **choose** an arbitrary value of θ as θ_1 and perform an experiment on the system $\mathcal{S}$ with $K(\theta_1)$. Define $\mathcal{T}_1 = \{\theta_1\}$ and $\mathcal{T}_1^{\tilde{V}} = \{\tilde{V}(\theta_1)\}$ where $\tilde{V}(\theta_1)$ is measured based on the collected input-output data during this experiment.
2. **for** $k = 1, ..., n_{it}$ **do**

2.1. **update** the mean $\mu_k(\theta)$ and the variance $\sigma_k^2(\theta)$ of the probabilistic model $\mathcal{V}_k(\theta)$ with the available data $\mathcal{T}_k$ and $\mathcal{T}_k^{\tilde{V}}$

2.2. **optimize** the acquisition function to determine θ_{k+1}

$$\theta_{k+1} \leftarrow \arg\max_{\theta \in \Theta} \mathcal{A}_k(\theta);$$

2.3. **execute** an experiment on $\mathcal{S}$ with $K(\theta_{k+1})$ and **measure** $\tilde{V}(\theta_{k+1})$;

2.4. **augment** the dataset:

$$\mathcal{T}_{k+1} \leftarrow \mathcal{T}_k \cup \theta_{k+1}$$

$$\mathcal{T}_{k+1}^{\tilde{V}} \leftarrow \mathcal{T}_k^{\tilde{V}} \cup \tilde{V}(\theta_{k+1})$$

3. **exit** when $k = n_{it}$;
4. **extract** the optimal controller parameters θ_{opt}, with

$$\theta_{opt} = \theta_{opt,n_{it}} = \arg\min_{\theta \in \mathcal{T}_{n_{it}}} \mu_{n_{it}}(\theta);$$

Output: Controller parameters θ_{opt}

B. Safe experimentation with constraints

As mentioned in Section II, we suppose that a model M of the uncertain dynamics $\mathcal{S}$ is available. Using this model M, let us define $V_M(\theta)$ as the performance level obtained with $K(\theta)$ when $\mathcal{S} = M$. In other words, when $V(\theta)$ is defined as the energy of the difference between the desired output and the output of the loop $[K(\theta)\ \mathcal{S}]$, $V_M(\theta)$ is defined as the energy of the difference between the desired output and the output of the loop $[K(\theta)\ M]$. In addition, we could also of course incorporate additional performance aspects in V_M (such as robustness conditions).

It is clear that, for a given θ, $V_M(\theta)$ can be computed either analytically or via simulation. This also means that this evaluation does not require an experiment on $\mathcal{S}$. If the model M is not a too poor representation of the unknown system $\mathcal{S}$, we can also say that, if a given value of θ leads to a (very) high value of $V_M(\theta)$ (i.e., the controller $K(\theta)$ achieves a (very) poor performance on M), we should avoid to make an experiment on the true system $\mathcal{S}$ with that controller $K(\theta)$.

This rationale has led us to the following modification of the optimization problem (1):

$$\arg\min_{\theta \in \Theta} V(\theta) \tag{4a}$$

$$\text{s.t. } V_M(\theta) < \varepsilon. \tag{4b}$$

where the set of θ such that $V_M(\theta) < \varepsilon$ represents the set of θ where the performance of the loop $[K(\theta)\ M]$ is acceptable. By acceptable, we mean that the performance

is not necessarily good, but the safety is ensured. In other words, $V_M(\theta) < \varepsilon$ is a safety constraint and we will denote $\Theta_{safe} = \{\theta \in \Theta \mid V_M(\theta) < \varepsilon\}$ the implicit set containing all parameter vectors θ leading to an a-priori safe experiment.

The constrained optimization problem (4) can also be treated using a BO algorithm. Due to the deterministic nature of the constraint in (4), one could use the BO formalism for such constraints. However, this leads to a prohibitive computing time due to the fact that the evaluation of the constraint cannot be easily *vectorized* when evaluating it for thousands of grid points. Consequently, we will here use a BO algorithm with a so-called *coupled constraint* [11], [12]. Even though this approach could seem less straightforward, it leads to satisfactory results as we will see in the next section.

When the constraint in (4) is treated as a coupled constraint, we determine, besides the GP model for $V(\theta)$, also a GP model for the computable quantity $V_M(\theta)$. In the Matlab implementation, the GP models of $V(\theta)$ and $V_M(\theta)$ are learned in a coupled manner. This means that, at Iteration k, the GP model for $V(\theta)$ is based on the data $\mathcal{T}_k$ and $\mathcal{T}_k^{\tilde{V}}$ and the GP model for $V_M(\theta)$ is based on the very same data $\mathcal{T}_k$ and $\mathcal{T}_k^{\tilde{V}}$. Consequently, since the GP model of $V_M(\theta)$ must represent this function over the whole set Θ, this particular *coupled* implementation will also require evaluations of $V(\theta)$ outside of the safety zone Θ_{safe} (and thus experiments on $\mathcal{S}$ with controllers $K(\theta)$ outside of this safety zone). This major drawback can however easily be avoided by slightly modifying the constrained optimization problem (4) to:

$$\arg\min_{\theta\in\Theta} J(\theta) \tag{5a}$$

$$\text{s.t. } V_M(\theta) < \varepsilon. \tag{5b}$$

where the objective function $J(\theta)$ (that replaces $V(\theta)$) is defined as follows:

$$J(\theta) = \begin{cases} V(\theta) & \text{if } V_M(\theta) < \varepsilon \\ V_M(\theta) & \text{if } V_M(\theta) \geq \varepsilon. \end{cases} \tag{6}$$

It is clear that the optimization problem (5) has the same solution as the optimization problem (4). In the sequel, the optimization problem (5) will be the one that we will solve using Bayesian Optimization to address our control design problem in a safe manner.

To solve (5) with a BO algorithm requires GP models of both $J(\theta)$ and $V_M(\theta)$. In order to explain why replacing $V(\theta)$ by $J(\theta)$ is useful in practice, let us first note that a GP model of $J(\theta)$ can easily be determined. In particular, at Iteration k, the GP model will be determined based on the data $\mathcal{T}_k$, $\mathcal{T}_k^{\tilde{J}} = \{\tilde{J}(\theta_1), ..., \tilde{J}(\theta_k)\}$ where, for $i = 1, ..., k$, $\tilde{J}(\theta_i) = \tilde{V}(\theta_i)$ when $V_M(\theta_i) < \varepsilon$ and $\tilde{J}(\theta_i) = V_M(\theta_i)$ when $V_M(\theta_i) \geq \varepsilon$. In other words, in order to deduce the GP model of $J(\theta)$, experiments on the system $\mathcal{S}$ will only be performed for parameter vectors $\theta_i \in \Theta_{safe}$ while, for the parameter vectors $\theta_i \notin \Theta_{safe}$, the measurement $\tilde{V}(\theta_i)$ that would entail a possibly dangerous experiment will be replaced by the performance $V_M(\theta_i)$ of the loop $[K(\theta_i)\ M]$ (that can be evaluated via simulation). This characteristic of the learning process of the GP model of $J(\theta)$ (which is, in the safety zone, a model of $V(\theta)$) allows us to learn the GP models of $J(\theta)$ and $V_M(\theta)$ over Θ without performing experiments on the system $\mathcal{S}$ for parameter vectors outside the safety zone Θ_{safe}.

Remark 1. With the GP models of $J(\theta)$ and $V_M(\theta)$, Algorithm 1 can also be used to solve (5) modulo two adaptations. First, in Step 2.2, the maximized quantity is changed to $\mathcal{A}_k(\theta)$ multiplied by the probability that the (current) GP model of $V_M(\theta)$ is smaller than ε [11]. Second, in Step 4, θ_{opt} is determined over the elements of $\mathcal{T}_{n_{it}}$ lying in Θ_{safe}.

Remark 2. In order to further simplify the optimization process, the optimization problem (5) could also be replaced by the unconstrained optimization problem $\arg\min_{\theta\in\Theta} J(\theta)$ with $J(\theta)$ as in (6). This removes the need of one of the two GP models (i.e., the one of $V_M(\theta)$). On the examples that will be presented in the sequel, this simplified optimization problem leads to very similar results as the ones with (5).

IV. Case studies

A. Linear example

We first consider a linear system $\mathcal{S}$ described by the following continuous-time transfer function $P(s)$ (s is the Laplace variable):

$$P(s) = \frac{10}{(s+10)(s+1)}. \tag{7}$$

For this linear system, we wish to determine a feedback controller $K(s, \theta)$ having the following structure:

$$K(s, \theta) = \frac{k(s+p_1)(s+p_2)}{s(s+4.2)},$$

where the three tunable parameters are gathered in the vector $\theta = (k, p_1, p_2)^T$. The control objective is to track a (unit) step reference with an overshoot of 5% and a rise time of one second. We thus consider the following reference model $M_{ref}(s) = 9/(s^2 + 4.2s + 9)$ whose step response determines the desired output $y_{des}(t)$ of the closed-loop system. It is clear that $y_{des}(t)$ can be obtained with a controller $K(s, \theta_{des,lin})$ where $\theta_{des,lin} = (0.9, 10, 1)^T$. However, this parameter vector $\theta_{des,lin}$ cannot be determined in practice since $P(s)$ is unknown.

In order to determine an optimal controller, we will use the BO approach presented in the previous sections. For this purpose, it is necessary to determine a cost function $V(\theta)$ whose minimization yields a parameter vector equal to (or at least as close as possible to) to $\theta_{des,lin}$ and to determine a type of experiments that allows, for any given value of θ, to collect input and output data with which an estimate $\tilde{V}(\theta)$ of this cost function $V(\theta)$ can be computed. The chosen experiment for a given θ here consists in applying, for five seconds, a unit step reference $c(t)$ to the closed loop

$[K(s,\theta)\ P(s)]$ and in collecting the corresponding (noisy) output data $y(t,\theta)$ at a sampling rate $T_s = 0.025\ s$:

$$\begin{cases} y(t,\theta) = P(s)u(t,\theta) + w(t) \\ u(t,\theta) = K(s,\theta)(c(t) - y(t)), \end{cases}$$

where $w(t)$ is the measurement noise (chosen here as a white noise of standard deviation 0.01). If we denote $y_n(\theta)$ $(n = 1, ..., 200)$ the sampled version of the continuous-time signal $y(t,\theta)$ and $y_{des,n}$ the one of $y_{des}(t)$, we can define $\tilde{V}(\theta) = T_s \sum_{n=1}^{200}(y_n(\theta) - y_{des,n})^2$. The corresponding cost function $V(\theta)$ is the expected value of $\tilde{V}(\theta)$ (note that we do not need to be able to compute $V(\theta)$, we only need to be able to compute $\tilde{V}(\theta)$).

As mentioned in the previous sections, we suppose that we have a model of the system $P(s)$. Here, this model $M(s)$ of $P(s)$ is given by

$$M(s) = \frac{12}{(s+8)(s+2)} \tag{8}$$

This model allows to determine the safety zone Θ_{safe} by choosing $\varepsilon = 0.1$ and by defining $V_M(\theta)$ as follows:

$$V_M(\theta) = \begin{cases} T_s \sum_{n=1}^{200} (y_{M,n}(\theta) - y_{des,n})^2 & \text{if } [K(s,\theta)\ M(s)] \text{ stable} \\ 100\ \varepsilon & \text{otherwise.} \end{cases} \tag{9}$$

with $y_{M,n}(\theta)$ the sampled version of $y_M(t,\theta) = \frac{M(s)K(s,\theta)}{1+M(s)K(s,\theta)} c(t)$.

The model $M(s)$ of $P(s)$ also allows to determine the parameter vector $\theta_{init} = (0.75, 8, 2)^T$ defining the controller $K(s,\theta_{init})$ which ensures that $y_M(t,\theta_{init})$ is exactly equal to $y_{des}(t)$ (i.e., $V_M(\theta_{init}) = 0$). Note that, when applied to $P(s)$, the performance of $K(s,\theta_{init})$ is nevertheless rather poor (see Figure 1). A BO algorithm aiming at solving (5) is thus useful to improve the performance.

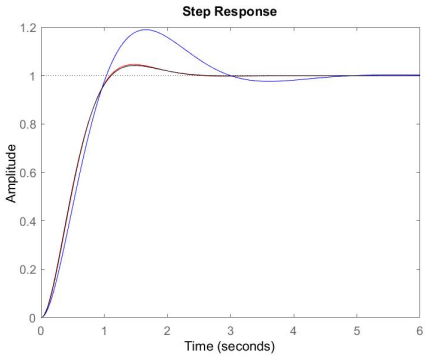

Fig. 1. $y_{des}(t)$ (red), $y(t,\theta_{opt,lin})$ (black) and $y(t,\theta_{init})$ (blue) for the linear case study.

With the above definitions, we perform $n_{it} = 300$ iterations of a BO algorithm aiming at solving (5). In order to initialize this BO algorithm, the first iteration will be performed with the parameter vector $\theta_1 = \theta_{init} = (0.75, 8, 2)^T$ (see Algorithm 1). This means that the first experiment on the system $P(s)$ will be performed with $K(s,\theta_{init})$. We also define the search space Θ based on θ_{init}: the first entry of θ can vary in $[0.05\ 1.75]$, the second entry in $[5\ 58]$ and the third entry in $[0.5\ 7]$.

These 300 iterations leads to $\theta_{opt,lin} = (0.98, 9.139, 1.009)^T$ which is clearly close to $\theta_{des,lin} = (0.9, 10, 1)^T$. As can be seen in Figure 1, the optimal output $y(t,\theta_{opt,lin})$ with that controller $K(s,\theta_{opt,lin})$ is very close to the desired $y_{des}(t)$ and the improvement with respect to $y(t,\theta_{init})$ is clear. The safe operation implemented in the optimization problem (5) has thus not prevented the determination of a good optimum. Let us now analyze the advantage of the safe operation. Among the 300 to-be-tested parameter vectors θ, only 269 lied in Θ_{safe} and have therefore been tested on the system $P(s)$. The sum of the cost $\tilde{V}(\theta)$ over these 269 values is equal to 5.83. Among the 31 parameter vectors θ for which $V_M(\theta) \geq 0.1$, five led to $K(s,\theta)$ destabilizing the model $M(s)$. If we would have applied the five controllers $K(s,\theta)$ destabilizing the model $M(s)$ on the true system $P(s)$, the sum of the cost $\tilde{V}(\theta)$ over these five experiments would have been 253 (to be compared to 5.83 for the 269 *good* parameter vectors $\theta \in \Theta_{safe}$). Note that this sum is finite because $\tilde{V}(\theta)$ is defined based on an experiment lasting only five seconds. If we would have applied the controller $K(s,\theta)$ with the 26 other parameter vectors θ such that $V_M(\theta) \geq 0.1$, the sum of $\tilde{V}(\theta)$ for these 26 experiments would have been of 11. So, the safe operation leads to a total cost of 5.83 and safes us a cost of 264. The total energy of the input $u(t)$ that is used during the 269 experiments is 1884 and, by not performing the 31 experiments with $\theta \notin \Theta_{safe}$, we save a total energy of 8750. The benefit of the safe operation is thus clear.

B. Nonlinear example

The previous control design problem could have been addressed using other techniques than a BO algorithm. We therefore now consider a nonlinear system $\mathcal{S}$ i.e., an Hammerstein system made up of an input saturation with limit $[-1.5,\ 1.5]$ and the transfer function $P(s)$ given in (7). Note that, if we apply either the initial controller $K(s,\theta_{init})$ or the controller $K(s,\theta_{opt,lin})$ determined in the previous subsection to this Hammerstein system, the input saturation will be activated (the control input indeed reaches 1.8 in the linear setup with both controllers).

In this nonlinear setup, $\tilde{V}(\theta)$ will have the same expression as in the previous subsection, but $y(t,\theta)$ will be determined in a closed loop made up of $K(s,\theta)$ and the Hammerstein system $\mathcal{S}$. The model M of $\mathcal{S}$ remains the linear transfer function $M(s)$ given in (8) and $V_M(\theta)$ also keeps the same expression (9). The desired output $y_{des}(t)$ is also kept unchanged. The same holds for the other settings of the BO algorithm aiming at solving (5).

After 300 iterations of this BO algorithm, we obtain $\theta_{opt,nl} = (1.737, 7.892, 0.5991)^T$. As can be seen in Figure 2, the optimal output $y(t,\theta_{opt,nl})$ with the controller $K(s,\theta_{opt,nl})$ is quite close to the desired $y_{des}(t)$ (but obviously less than in the linear case) and the improvement

with respect to θ_{init} is here also clear. It is also to be noted that $K(s, \theta_{opt,nl})$ could not be determined in an obvious way from the control specifications and that, as shown in Figure 2, it clearly outperforms the controller $K(s, \theta_{des,lin})$ i.e., the controller for which $y(t, \theta_{des,lin}) = y_{des}(t)$ when $\mathcal{S} = P(s)$ (and the noise $w(t) = 0$).

We have seen that the safe operation has thus here also not prevented the determination of a good optimum. Let us now analyze the advantage of the safe operation. Among the 300 to-be-tested parameter vectors θ, only 261 lied in Θ_{safe} and have therefore been really tested on the Hammerstein system $\mathcal{S}$. The sum of the cost $\tilde{V}(\theta)$ over these 261 values is equal to 6.47. Among the 39 parameter vectors θ for which $V_M(\theta) \geq 0.1$, five led to $K(s, \theta)$ destabilizing the model $M(s)$. If we would have applied these 39 parameter vectors to the Hammerstein system, the sum of $\tilde{V}(\theta)$ would have been of 8.4. With respect to the linear case, this figure is here smaller since the parameter vectors θ leading to controllers destabilizing $M(s)$ do not lead to a fast divergence of $y(t, \theta)$ when applied to the Hammerstein system (we obtain a sort of limit cycle; see Figure 3). The total energy of the input $u(t)$ that is used during the 261 experiments with $\theta \in \Theta_{safe}$ is 1701 and, by not performing the 39 experiments with $\theta \notin \Theta_{safe}$, we save a total energy of 290. The benefit of the safe operation is thus still there.

As mentioned in the previous paragraph, five values of θ have been disregarded because $K(s, \theta)$ destabilizes $M(s)$. One of these values is $\theta_{uns} = (0.16, 48.5, 4.2)^T$. The output $y(t, \theta_{uns})$ obtained by applying $K(s, \theta_{uns})$ on the Hammerstein system $\mathcal{S}$ is represented in Figure 3 for a simulation of 30 seconds. The output $y(t, \theta_{uns})$ seems to reach a sort of limit cycle and thus does not diverge, but it is clear that this experiment is better not done in practice (as allowed by our safe operation). We observe a similar behaviour for the other four values of θ that have been disregarded because $K(s, \theta)$ destabilizes $M(s)$.

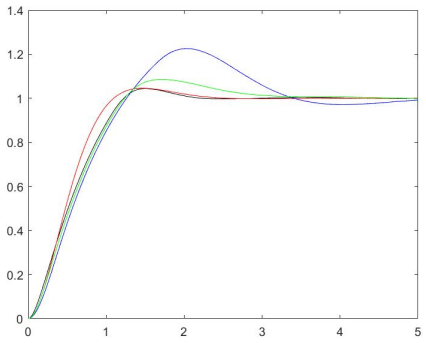

Fig. 2. $y_{des}(t)$ (red), $y(t, \theta_{opt,nl})$ (black), $y(t, \theta_{init})$ (blue) and $y(t, \theta_{des,lin})$ (green) for the nonlinear case study.

V. Conclusions

We have presented a controller calibration procedure based on Bayesian Optimization, where repeated experiments are performed on the true system aiming at minimization of a given closed-loop control objective. Compared to existing techniques, we introduce explicit constraints for safe exploration of the controller parameter space, exploiting knowledge from an available process model. Numerical examples showcase the advantage of the proposed methodology. Current and future research is aimed at the use of robust control analysis and design tools to further improve the safety constraint imposed to the BO algorithm.

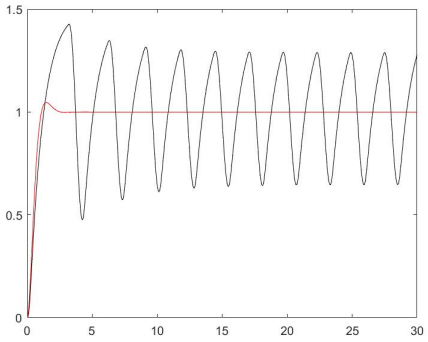

Fig. 3. $y_{des}(t)$ (red), $y(t, \theta_{uns})$ (black).

References

[1] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[2] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 491–496.

[3] L. Roveda, M. Forgione, and D. Piga, "Robot control parameters auto-tuning in trajectory tracking applications," *Control Engineering Practice*, vol. 101, p. 104488, 2020.

[4] M. Neumann-Brosig, A. Marco, D. Schwarzmann, and S. Trimpe, "Data-efficient autotuning with bayesian optimization: An industrial control study," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 730–740, 2019.

[5] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[6] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[7] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," in *International conference on machine learning*. PMLR, 2015, pp. 997–1005.

[8] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics," *Machine Learning*, pp. 1–35, 2021.

[9] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration for interactive machine learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[10] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006, vol. 2, no. 3.

[11] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints." in *ICML*, vol. 2014, 2014, pp. 937–945.

[12] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," *arXiv preprint arXiv:1403.5607*, 2014.