Virtual vs. Real: Trading Off Simulations and Physical Experiments in Reinforcement Learning with Bayesian Optimization

Alonso Marco[1,5], Felix Berkenkamp[2,5], Philipp Hennig[1,5], Angela P. Schoellig[3], Andreas Krause[2,5], Stefan Schaal[1,4,5], and Sebastian Trimpe[1,5]

Abstract— **In practice, the parameters of control policies are often tuned manually. This is time-consuming and frustrating. Reinforcement learning is a promising alternative that aims to automate this process, yet often requires too many experiments to be practical. In this paper, we propose a solution to this problem by exploiting prior knowledge from simulations, which are readily available for most robotic platforms. Specifically, we extend Entropy Search, a Bayesian optimization algorithm that maximizes information gain from each experiment, to the case of multiple information sources. The result is a principled way to automatically combine cheap, but inaccurate information from simulations with expensive and accurate physical experiments in a cost-effective manner. We apply the resulting method to a cart-pole system, which confirms that the algorithm can find good control policies with fewer experiments than standard Bayesian optimization on the physical system only.**

I. Introduction

Typically, the control policies that are used in robotics depend on a small set of tuning parameters. To achieve the best performance on the real system, these parameters are usually tuned manually in experiments on the physical platform. Policy search methods in reinforcement learning aim to automate this process [1]. However, without prior knowledge, these methods can require significant amounts of experimental time before determining optimal, or even only reasonable parameters. In robotics, simulation models of the robotic system are usually available, e.g., as a by-product of the design process. While exploiting knowledge from simulation models has been considered before, no principled way to trade off between the relative costs and accuracies of simulations and experiments exists [2]. As a result, state-of-the-art reinforcement learning methods require more experimental time on the real system than necessary.

In this paper, we propose a new reinforcement learning method that can automatically optimize the parameters of control policies based on data from different information sources, such as simulations and experiments. Specifically, we use an extension of Entropy Search [3], [4], a Bayesian optimization framework for information-efficient global optimization. The resulting method automatically trades off the amount of information gained from different sources with their respective costs and requires fewer physical experiments to determine optimal parameters (see Fig. 1).

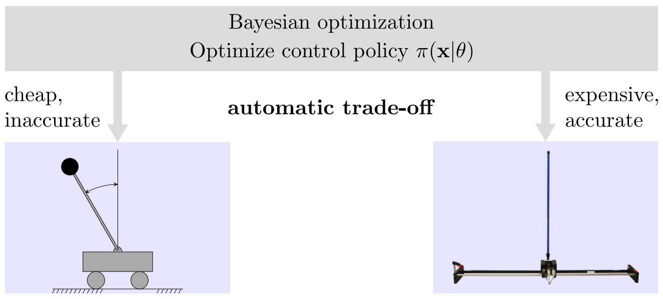


Fig. 1. The proposed algorithm optimizes the parameters $\boldsymbol{\theta}$ of a control policy based on data of a cheap, but inaccurate simulation and expensive data from the real system. By actively trading off between the information that can be gained from each system relative to their costs, the algorithm requires significantly fewer evaluations on the physical system.

Related work: Improving the performance of reinforcement learning with prior model information from a simulator has been considered before. A typical approach is two-stage learning, where algorithms are trained for a certain amount of time in simulation in order to warm-start the learning on the real robot [2]. For example, [5] reports performance improvements when using model information from simulation as a prior for real experiments. Transfer learning is a similar approach that aims to generalize between different tasks, rather than from a simulated model to the real system [6]. The work in [7] learns an optimal policy and value function of a finite Markov decision process based on models with different accuracies. They rely on hierarchical models and switch to higher accuracy models once a threshold accuracy has been reached at a lower level. A commonly used reinforcement learning method is policy gradients [8], where policy parameters are improved locally along the gradient. In this setting, simulation knowledge can be used to estimate the gradient of real experiments [9]. However, policy gradient methods only converge to locally optimal parameters. None of the above methods explicitly considers the cost of experiments on a robot. In this paper, we actively trade off the different costs and information gains associated with simulation and real experiments and obtain

[1] Max Planck Institute for Intelligent Systems, Tübingen, Germany. Email: <firstname>.<lastname>@tuebingen.mpg.de
[2] Learning & Adaptive Systems Group (LAS), Department of Computer Science, ETH Zurich, Switzerland. Email: {befelix, krausea}@ethz.ch
[3] Dynamic Systems Lab (DSL), University of Toronto Institute for Aerospace Studies (UTIAS), Canada. Email: schoellig@utias.utoronto.ca
[4] Computational Learning and Motor Control Lab, University of Southern California, USA.
[5] Max Planck ETH Center for Learning Systems, Tübingen, Germany, and Zürich, Switzerland.
This research was supported in part by the Max Planck ETH Center for Learning Systems, the Max Planck Society, SNSF grant 200020_159557, NSERC grant RGPIN-2014-04634, the Connaught New Researcher Award, and National Science Foundation grants IIS-1205249, IIS-1017134, EECS-0926052, the Office of Naval Research, the Okawa Foundation

globally optimal parameter estimates.

A method that has been particularly successful for parameter optimization in robotics is Bayesian optimization [10]. In particular, methods based on Gaussian process (GP, [11]) models are widely used because of their ability to determine globally optimal parameters within few evaluations. Examples include gait optimization in legged robots [12] and controller optimization for a snake-like robot [13]. In [14], the controller parameters of a linear state-feedback controller were optimized using the LQR framework as a low-dimensional representation of controller policies, while in [15] the control policy itself was defined by Bayesian optimization with a specifically chosen kernel. Safety constraints on the robot during the optimization process were considered in [16]. A comparison of different Bayesian and non-Bayesian global optimization methods can be found in [17]. All the previous methods use Bayesian optimization directly on the real system. In contrast, we consider an extension of Bayesian optimization that can extract additional information from a simulator and speed up the optimization process.

The methodology herein is related to multi-task Bayesian optimization, where one aims to transfer knowledge about two related tasks [18], [19]. A GP model with multiple information sources was first considered in [20]. Since then, optimization with multiple information sources has been considered under strict requirements, such as models forming a hierarchy of increasing accuracy and without considering different costs [21], [22]. More recently, [23] used a myopic policy, called the 'knowledge gradient' by [24], in order determine, which parameters to evaluate.

Our contribution: In this paper, we present a Bayesian optimization algorithm for multiple information sources. We use entropy [3], [4] to measure the information content of simulations and experiments. Since this is an appropriate unit of measure for the utility of both sources, our algorithm is able to compare physically meaningful quantities in the same units on either side, and trade off accuracy for cost. As a result, the algorithm can automatically decide whether to evaluate cheap, but inaccurate simulations or perform expensive and precise real experiments. We apply the method to optimize the policy of a cart-pole system and show that this approach can speed up the optimization process significantly compared to standard Bayesian optimization [14]. The main contributions of the paper are *(i)* a novel Bayesian optimization algorithm that can trade off between costs of multiple information sources and *(ii)* the first application of such a framework to the problem of reinforcement learning and optimization of controller parameters.

For convenience within the next sections, we rename the concepts *accuracy* and *cost*: We now refer to the lack of accuracy of a controller as *cost*. Furthermore, we now denominate the cost of retrieving an evaluation from a specific information source as *effort*.

II. Problem Statement

We consider a reinforcement learning setting, where we aim to find an optimal policy to complete a certain task on a dynamic system. While we do not have access to a perfect model of the system, we assume that a control policy is available, which is parameterized by parameters $\boldsymbol{\theta}$ within some domain $\mathcal{D}$. The goal is to determine the optimal parameters $\boldsymbol{\theta}_{\min}$ that globally minimize the cost of a task,

$$\boldsymbol{\theta}_{\min} \in \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathcal{D}} J(\boldsymbol{\theta}). \tag{1}$$

The cost $J(\boldsymbol{\theta})$ measures the performance of the policy on a certain task on the real system. For example, one evaluation of the cost function could consist of controlling a robot with the parameterized policy and measuring an error signal over a fixed time horizon. To solve the optimization problem in (1), we can query a parameter vector $\boldsymbol{\theta}_n$ at each iteration n and observe the resulting performance $J(\boldsymbol{\theta}_n)$. Since these experiments cause wear in the robot and take time, the goal is to minimize the number of iterations before the optimal parameters in (1) are determined.

We assume that a simulation of the system is available, which we want to exploit to solve (1) more efficiently with fewer evaluations on the physical system. While simulations are only an approximation of the real world and cannot be used to determine the optimal parameters in (1) directly, they can be used to provide an estimate $J_{\text{sim}}(\boldsymbol{\theta})$ of the true cost. We use this estimate to obtain information about the location of the optimal parameters on the real system. As a result, at each iteration n, we do not only choose the next parameters $\boldsymbol{\theta}_n$ to evaluate, but also whether to perform a simulation or an experiment.

Both experiments, in the real world and in simulation, have physically meaningful evaluation efforts associated to them. For example, the effort may account for the amount of time required to complete an experiment/simulation and for monetary costs such as wear in the system. The overall goal is to minimize the total effort incurred in the experiments and simulations until the optimal parameters (1) on the real system are found.

III. Preliminaries

We start by introducing the necessary background information on GPs and Bayesian optimization.

A. Gaussian Processes (GPs)

While the cost $J(\boldsymbol{\theta})$ in (1) can easily be evaluated in an experiment for a given parameter $\boldsymbol{\theta}$, the functional relationship between parameters and the cost is unknown *a priori*. We use GPs as a nonparametric model to approximate the unknown function. The goal is to find an approximation of the nonlinear map, $J(\boldsymbol{\theta})\colon \mathcal{D} \mapsto \mathbb{R}$, from an input vector $\boldsymbol{\theta} \in \mathcal{D}$ to the function value $J(\boldsymbol{\theta})$. This is accomplished by modeling function values $J(\boldsymbol{\theta})$, associated with different values of $\boldsymbol{\theta}$, as random variables so that any finite number of these random variables have a joint Gaussian distribution [11].

For the nonparametric regression, we define a prior mean function $m(\boldsymbol{\theta})$, which encodes prior knowledge about the function $J(\cdot)$, and a covariance function $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$, which defines the covariance of any two function values, $J(\boldsymbol{\theta})$ and $J(\boldsymbol{\theta}')$, $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{D}$, and is used to model uncertainty about the mean estimate. The latter is also known as the kernel. The choice of kernel is problem-dependent and encodes assumptions about smoothness and rate of change of the unknown function, $J(\cdot)$.

The GP framework can be used to predict the function value $J(\boldsymbol{\theta}^*)$ at an arbitrary input $\boldsymbol{\theta}^* \in \mathcal{D}$, based on a set of n past observations $\mathcal{D}_n = \{\boldsymbol{\theta}_i, \hat{J}(\boldsymbol{\theta}_i)\}_{i=1}^n$. We assume that observations are noisy measurements of the true function; that is,

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \omega(\boldsymbol{\theta}), \tag{2}$$

where the noise $\omega(\boldsymbol{\theta}) \sim \mathcal{N}(0, \eta^2(\boldsymbol{\theta}))$ depends on the input. Conditioned on the previous observations, the mean and variance of the posterior normal distribution are

$$\mu_n(\boldsymbol{\theta}^*) = m(\boldsymbol{\theta}^*) + \mathbf{k}_n(\boldsymbol{\theta}^*)\mathbf{K}_n^{-1}\hat{\mathbf{y}}_n, \tag{3}$$

$$\sigma_n^2(\boldsymbol{\theta}^*) = k(\boldsymbol{\theta}^*, \boldsymbol{\theta}^*) - \mathbf{k}_n(\boldsymbol{\theta}^*)\mathbf{K}_n^{-1}\mathbf{k}_n^{\mathrm{T}}(\boldsymbol{\theta}^*), \tag{4}$$

where $\hat{\mathbf{y}}_n = \left[\hat{J}(\boldsymbol{\theta}_1) - m(\boldsymbol{\theta}_1), \ldots, \hat{J}(\boldsymbol{\theta}_n) - m(\boldsymbol{\theta}_n)\right]^{\mathrm{T}}$ is the vector of observed, noisy deviations from the mean, the vector $\mathbf{k}_n(\mathbf{a}^*) = \left[k(\boldsymbol{\theta}^*, \boldsymbol{\theta}_1), \ldots, k(\boldsymbol{\theta}^*, \boldsymbol{\theta}_n)\right]$ contains the covariances between the new input $\boldsymbol{\theta}^*$ and the observed data points in $\mathcal{D}_n$, and the symmetric matrix $\mathbf{K}_n \in \mathbb{R}^{n \times n}$ has entries $[\mathbf{K}_n]_{(i,j)} = k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) + \delta_{ij}\eta^2(\boldsymbol{\theta}_i)$, $i, j \in \{1, \ldots, n\}$.

B. Bayesian Optimization

We want to use the GP model of the cost function for parameter optimization. Using statistical models of an objective function for optimization is known as Bayesian optimization [10] in the literature. It comprises a class of data-efficient optimization methods that aim to determine the global optimum of cost functions that are expensive to evaluate. In our case, each evaluation of the cost with certain controller parameters on the robot cause wear in the system and may take a long time to perform. In the case of GP models, the mean, (3), and variance, (4) can be used to determine new parameters to evaluate that are promising candidates for the global optimum. For example, [25] uses upper confidence bounds that allow for provable convergence guarantees.

In this paper, we build upon the Entropy Search (ES, [3]) algorithm, which selects parameters in order to maximally reduce the uncertainty about the location of the minimum of $J(\boldsymbol{\theta})$ in each step. It quantifies this uncertainty through the entropy of the distribution over the location of the minimum,

$$p_{\min}(\boldsymbol{\theta}) = \mathbb{P}\big(\boldsymbol{\theta} \in \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathcal{D}} J(\boldsymbol{\theta})\big). \tag{5}$$

The approach becomes tractable by approximating $p_{\min}$ on a non-uniform grid, with higher resolution in areas where it is more likely to find the minimum. The key idea is that, upon convergence, we expect $p_{\min}$ to be peaked around the minima, thus to have low entropy. The rate of change in the entropy of $p_{\min}$ determines how much information about the location of the global minimum we obtain with each evaluation of the cost function. Given this metric, the optimal parameter at which to evaluate the cost function in the next iteration, is the one that is most informative:

$$\boldsymbol{\theta}_{n+1} = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{D}} \mathrm{E}\left[\Delta H(\boldsymbol{\theta})\right], \tag{6}$$

where $\Delta H(\boldsymbol{\theta})$ is the change in entropy of $p_{\min}$ caused by retrieving a new cost value at location $\boldsymbol{\theta}$. Intuitively, by collecting cost values at the most informative locations (6), we keep decreasing the amount of entropy in $p_{\min}$ until eventually $p_{\min}$ is peaked around the optima. At iteration n, we compute the best guess $\boldsymbol{\theta}_{\mathrm{bg}}$ about the optimal parameters by minimizing the current GP posterior (3):

$$\boldsymbol{\theta}_{\mathrm{bg}} = \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathcal{D}} \mu_n(\boldsymbol{\theta}). \tag{7}$$

The computation of ΔH given the GP model of $J(\cdot)$ requires several approximations. A full derivation is beyond the scope of the paper, but all details can be found in [3].

IV. Reinforcement Learning with Simulations

In this section, we show how the ES algorithm can be extended to multiple sources of information, such as simulations and physical experiments. The two main challenges are modeling the errors of the simulator in a principled way and trading off evaluation effort and information gain. In the following, we focus on the case where only one simulation is available for ease of exposition. However, the approach can easily be extended to an arbitrary number of information sources.

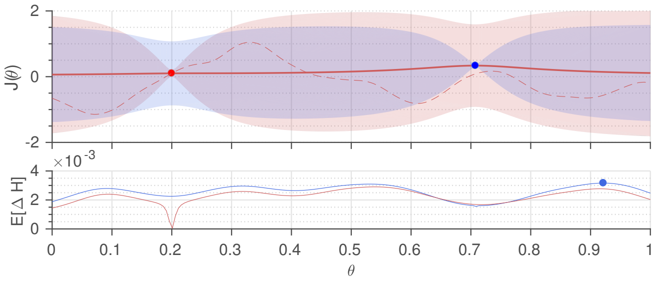
A. GP Model for Multiple Information Sources

To model the choice between simulation and phisical experiment, we use a specific kernel structure that is similar to the one used in [23]. The key idea is to model the cost on the real system as being partly explained through the simulator plus some error term. That is, $J(\boldsymbol{\theta}) = J_{\mathrm{sim}}(\boldsymbol{\theta}) + J_{\mathrm{err}}(\boldsymbol{\theta})$, where the true cost consists of the estimated cost of the simulation, $J_{\mathrm{sim}}(\boldsymbol{\theta})$, and a systematic error term, $J_{\mathrm{err}}(\boldsymbol{\theta})$. To incorporate this in the GP framework of Sec. III-A, we extend the parameter vector by an additional binary variable δ, which indicates whether the cost is evaluated in simulation ($\delta = 0$) or on the physical system ($\delta = 1$). Based on the extended parameter $\mathbf{a} = (\boldsymbol{\theta}, \delta)$, we can model the cost by adapting the GP kernel to

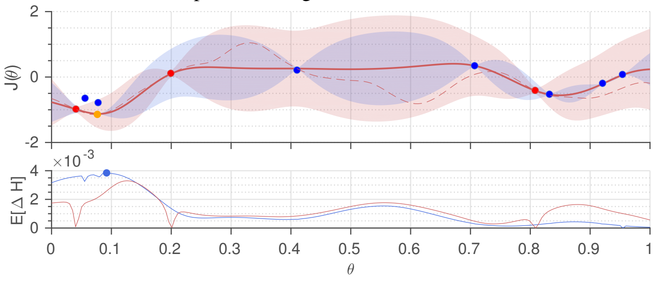
$$k(\mathbf{a}, \mathbf{a}') = k_{\mathrm{sim}}(\boldsymbol{\theta}, \boldsymbol{\theta}') + k_{\delta}(\delta, \delta')\, k_{\mathrm{err}}(\boldsymbol{\theta}, \boldsymbol{\theta}'). \tag{8}$$

The kernels $k_{\mathrm{sim}}(\cdot,\cdot)$ and $k_{\mathrm{err}}(\cdot,\cdot)$ model the cost function on the simulator and its difference to the cost on the physical system, respectively. The kernel $k_{\delta}(\delta, \delta') = \delta\delta'$ is equal to one if both parameters indicate a physical experiment and zero otherwise.

From Sec. III-A, we know that the kernel (8) models the covariances for different parameters. Intuitively, the kernel (8) encodes that two experiments on the physical system covary strongly. However, if one of the δ-variables is zero (i.e., a simulation), then the only covariances between the two values is captured by k_{sim}. Effectively, the error covariance is switched off in simulations in order to model that simulations



a: Exploration stage after two evaluations



b: Exploration stage after ten evaluations

Fig. 2. Synthetic example of how simulations and physical experiments can be combined by trading off information and evaluation effort. In (a), top, it is shown the GP posterior conditioned on one simulation (blue dot) and one physical experiment (red dot). The GP model from Sec. IV-A encodes that a portion of the uncertainty in the cost of the real system (red shaded) can be explained through the simulator (blue shaded). The red dashed line represents the cost function of the physical system. The cost function of the simulator is omitted for simplicity. In (a), bottom, it is shown the expected information gain per unit of effort of the simulator (blue line), and of the physical system (red line). The most informative point (blue dot) is selected among the two sources by the proposed method as next evaluation (in this case, a simulation). In (b), top, it is shown the GP posterior after nine iterations. The global minimum (orange dot) is found close to the true minimum.

cannot provide all the information about J. By choosing the kernels k_{sim} and k_{err}, we can model to what extend J can be explained by the simulator and thereby its quality. This is illustrated with a synthetic example in Fig. 2. The total variance of the cost on the physical system is shown in red. Before any data is observed, it is equal to the uncertainty about the simulation plus the uncertainty about the error. As shown in Fig. 2a, the blue shaded region highlights the variance of the simulator. Evaluations in simulation (blue dots) reduce the uncertainty of this blue shaded region, but reduce only partially the uncertainty about the true cost (red). In contrast, an evaluation on the real system (red dot) allows one to learn the true cost J directly, thus reducing the total uncertainty (red), while some uncertainty about the variance of J_{sim} remains (blue). Having uncertainty about the simulation is by itself irrelevant for the proposed method, because we solely aim to minimize the performance on the physical system.

Next to the kernel, we account for different amounts of noise in simulation (typically noise-free) and on the real system. That is, the noise variance of measurements, $\eta^2(\mathbf{a})$ in (2), takes different values, η^2_{exp} and η^2_{sim}, depending on whether an experiment or a simulation is chosen. With this kernel and noise structure, the two information sources can be modeled by a single GP, and predictions can be made according to (3) and (4).

B. Optimization

With the GP model defined, we now consider how it can be used to trade off accuracy for evaluation effort. As a first step, we quantify the goal. As before, we want to minimize the cost (1) on the real system. This means, the distribution over the minimum is defined in terms of the same cost (5) as in standard ES. In order to approximate p_{min}, we need to use the GP kernel with the additional δ factor fixed to one,

$$p_{\text{min}}(\boldsymbol{\theta}) = \mathbb{P}\big(\boldsymbol{\theta} \in \underset{\boldsymbol{\theta}\in\mathcal{D},\,\delta=1}{\operatorname{argmin}}\; J(\boldsymbol{\theta},\delta)\big). \tag{9}$$

As in ES, the goal is to arrive at a distribution p_{min} that has low entropy (i.e., very peaked on a certain location). The expected change in entropy is an appropriate measure for this. However, this quantity additionally depends on the variable δ, so that the algorithm has an additional degree of freedom in the parameters to optimize. If one were to use the same optimization problem as in (6), the algorithm would always choose to evaluate parameters with $\delta = 1$. This is because the experiments with $\delta = 1$ provide information about the cost function J directly, while an evaluation with $\delta = 0$ only provides information about part of the cost, J_{sim}.

To trade off between the two choices more appropriately, we associate an effort measure with both kinds of evaluations; t_{sim} for the simulation and t_{exp} for physical experiments. While simulations are less informative about p_{min}, they are significantly cheaper than experiments on a physical platform so that $t_{\text{sim}} < t_{\text{exp}}$. These effort measures can have physically meaningful units, such as the amount of time taken by a simulation relative to a physical experiment. While the effort measures are important to trade off the relative gains in information, they do not require tuning. For example, setting the effort of the simulator too high may lead to more experiments on the physical system than necessary, but the optimal parameters on the real system are found regardless.

A key advantage of using entropy to determine progress towards the goal is that it is a consistent unit of measurement for both information sources, even in the case of different noise variances. As a result, we can compare the gain in information about the location of the minimum (i.e., p_{min}) in simulation and physical experiments relative to their efforts. Thus, we select the next parameters, $\boldsymbol{\theta}_{n+1}$, and where to evaluate them, δ, according to

$$\underset{\boldsymbol{\theta}\in\mathcal{D},\, i\in\{\text{sim},\text{exp}\}}{\operatorname{argmax}} \quad \mathrm{E}\left[\Delta H_i(\boldsymbol{\theta})\right] / t_i. \tag{10}$$

The expected gain in entropy, $\mathrm{E}\left[\Delta H_i\right]$, depends on whether we evaluate in simulation or physical experiment. By selecting the best gain per unit of effort, the algorithm automatically decides which kind of evaluation decreases the uncertainty about the location of the minimum the most, relative to effort. Importantly, since the GP model in (8) is adaptive to the quality of the simulator, the acquisition function (10) leads to informed decisions about whether the simulator is reliable enough to lead to additional information.

We illustrate a typical run of the algorithm in Fig. 2. The algorithm was initialized with one physical experiment (red dot in Fig. 2a) for the purpose of illustration. The evaluation effort of the simulator was set to 40% less of that of the real system. As a result, it is advantageous to exploit initially the low effort that takes to do simulations. The algorithm automatically decides to do so, as can be seen in Fig. 2a. The simulation (blue dot) decreases the amount of uncertainty about the simulation model, but provides only partial information about the true cost of the system. As a result, the method eventually starts to evaluate parameters on the real system. Notice that this is not the same as two stage learning, because the algorithm can decide to switch back to simulations if this is beneficial. This is especially important in situations where the quality of the simulation is not known in advance and the hyperparameters of the kernels in (8) are optimized. Eventually, the algorithm converges to a distribution $p_{\min}$ that is peaked around the minima of the cost function. Since the model can exploit cheap information from simulation, fewer physical experiments are needed to determine the minimum than if only physical experiments were used.

Because the proposed method extends Entropy Search (ES) to multiple information sources, we refer to it as *Multi-fidelity Entropy Search* (MF-ES).

V. Experimental Results

In this section, we evaluate MF-ES for optimizing the feedback controller of an unstable cart-pole system, as illustrated in Fig. 1.

A. Experimental Setup

As experimental setup, we use the Quanser Linear Inverted Pendulum, [26]. The dynamics of the system are described by

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, u_k), \tag{11}$$

where $\mathbf{x}_k = [s_k, \psi_k, \dot{s}_k, \dot{\psi}_k]^{\mathrm{T}}$ is the state at discrete time step k, which is comprised of pendulum angle ψ, cart position s, and their time derivatives; u_k is the commanded motor voltage driving the cart; and $\mathbf{f}(\cdot)$ is the transition function (see [26] for details).

The cart-pole setup is connected through dedicated hardware to a standard Laptop and can be controlled via Matlab/Simulink. A nonlinear Simulink model of the system dynamics (11) is provided by the manufacturer and used as the simulator in our setting.

B. Controller Tuning Problem

To stabilize the pendulum about its upright equilibrium, we use a static state-feedback controller,

$$u_k = \mathbf{F}\mathbf{x}_k, \tag{12}$$

with gain matrix $\mathbf{F} \in \mathbb{R}^{1\times 4}$. We seek optimal gains $\boldsymbol{F}$ that minimize the cost function

$$J = \frac{1}{K}\sum_{k=0}^{K-1} s_k^2 + \psi_k^2 + \dot{s}_k^2 + 0.1\dot{\psi}_k^2 + 10^{-1.5}u_k^2 \tag{13}$$

over a sufficiently long time horizon K. The cost (13) penalizes deviations from the equilibrium $\mathbf{x} = 0$ and control effort (u_k^2).

Instead of tuning the controllers gains $\mathbf{F}$ directly (i.e. setting $\boldsymbol{\theta} = \mathbf{F}$), we follow the approach from [14], [27], and pre-structure suitable controllers gains by means of a Linear Quadratic Regulator (LQR, [28]) design using a nominal, *linearized* version, $(\mathbf{A}, \mathbf{B})$, of the dynamics in (11), around the aforementioned equilibrium, which can be obtained from the simulator, for example. That is, the controller gain is computed from a discrete-time LQR design,

$$\mathbf{F} = \mathrm{dlqr}(\mathbf{A}, \mathbf{B}, \mathbf{W}_{\mathrm{x}}(\boldsymbol{\theta}), \mathbf{W}_{\mathrm{u}}(\boldsymbol{\theta})), \tag{14}$$

where $\mathbf{W}_{\mathrm{x}}(\boldsymbol{\theta})$ and $\mathbf{W}_{\mathrm{u}}(\boldsymbol{\theta})$ are suitable parameterizations of LQR weights (see [14], [27] for details). Here, we selected

$$\mathbf{W}_{\mathrm{x}}(\boldsymbol{\theta}) = \mathrm{diag}(10^{\theta_1}, 1, 1, 0.1), \qquad \theta_1 \in [-3, 2], \tag{15}$$

$$\mathbf{W}_{\mathrm{u}}(\boldsymbol{\theta}) = 10^{-\theta_2}, \qquad \theta_2 \in [1, 5]. \tag{16}$$

Hence, we are left with tuning two parameters, $\boldsymbol{\theta} \in \mathbb{R}^2$.

Advantages of the LQR parameterization in (14) are the possibility of dimensionality reduction, exploitation of prior knowledge in form of a linear dynamics model, and guarantees of stability and certain robustness properties with respect to the nominal system (see [14] for a discussion). However, we emphasize that LQR-weights are only one possible way to parameterize feedback controllers; alternative parameterizations [29] or direct tuning of the gains $\mathbf{F}$ [16] is also possible. The method proposed herein is independent of the specific parameterization used.

With the above definitions, the cost function $J(\boldsymbol{\theta})$, which we seek to minimize (1), is defined by equations (13)–(16). An evaluation of the cost $J_{\exp}(\boldsymbol{\theta}^*)$ is obtained by computing the controller gain (14) based on the weight matrices (15), (16), performing a 30 s balancing experiment on the physical system, and computing the cost according to (13) from the experimental data $\{\mathbf{x}_k, u_k\}_{k=0,\ldots,K-1}$. A simulation sample $J_{\mathrm{sim}}(\boldsymbol{\theta}^*)$ is obtained in the same manner by running a 30 s simulation instead.

If a candidate controller violates safety limits on the states and inputs, it is determined as *unstable*, and we assign a fixed penalty of $J_{\exp} = 0.06$ and $J_{\mathrm{sim}} = 0.04$ for physical experiment and simulation, respectively. These numbers are chosen conservatively larger than the cost of the worse stabilizing controller observed after some a priori initial evaluations. Thus, evaluations during the learning procedure shall not result in higher costs than these.

The controller is automatically tuned over roll-outs without human intervention. To this end, a nominal[1] controller $\boldsymbol{\theta}_{\mathrm{nom}} = [0,\ 1.5]$ is balancing the pole when no tuning experiment is being performed. The optimizer triggers new experiments, when an evaluation on the real system is required. As soon as the experiment is finished, or instability is

[1]The nominal controller is the optimal controller if the true dynamics was linear according to the nominal model $(\mathbf{A}, \mathbf{B})$. Then, choosing $\mathbf{W}_{\mathrm{x}}(\boldsymbol{\theta})$ and $\mathbf{W}_{\mathrm{u}}(\boldsymbol{\theta})$ corresponding to the cost (13) yields the optimal controller $\mathbf{F}$, see [14]. This is a typical choice when neglecting the nonlinear dynamics.

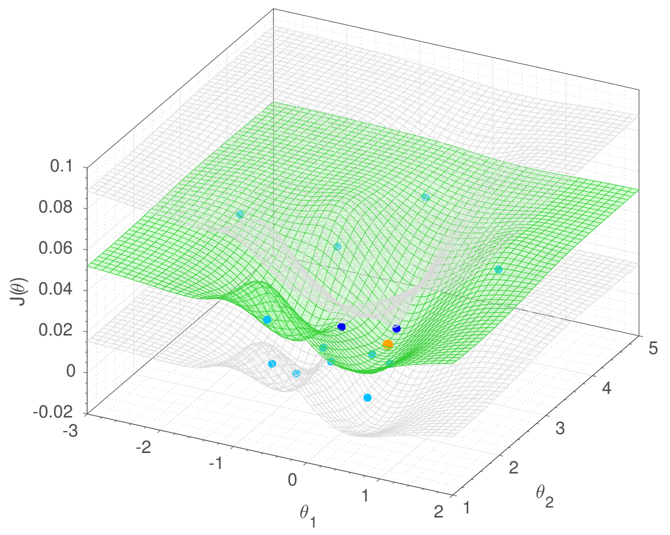


Fig. 3. GP posterior after termination of the exploration with MF-ES. The evaluations on the simulator (light blue) are systematically below the evaluations on the real system (dark blue). This bias is captured by the GP model assuming a lower prior mean for the simulator data, as mentioned in Sec. V-C. The posterior mean (green surface) and ± 2 std (grey surface) predict the underlying cost function of the real system, conditioned on the observed data from both simulator and experiments. The best guess location for the global minimum, $\boldsymbol{\theta}_{\text{bg}}$, is represented by the orange dot.

detected, the system switches back to the nominal controller. The nominal controller shows very poor performance, which shall be improved with the proposed RL method.

C. Bayesian Optimization Settings

We apply the method of Sec. IV, MF-ES, to optimize the experimental cost (13) by querying simulations and experiments. The efforts in (10) correspond to the approximate times we need to wait until a simulation is computed and a physical experiment is performed, $t_{\text{sim}} = 1\,\text{s}$ and $t_{\text{exp}} = 30\,\text{s}$, i.e., simulations require 30 times less effort than physical experiments.

For the GP model, we choose the rational quadratic kernel with $\alpha = 1/4$ (see [11]) for both k_{sim} and k_{err} in (8). Hyperparameters, such as length scales and output variances, were chosen from some initial experiments and then held fixed during optimization. As prior mean functions, we use $m_{\text{sim}}(\boldsymbol{\theta}) \equiv 0.04$ and $m_{\text{err}}(\boldsymbol{\theta}) \equiv 0.02$, respectively, for the simulation and error GP. These choices correspond to the penalties J_{sim} and J_{exp} given for unstable controllers (adding m_{sim} and m_{err} for the experiment). Hence, the prior mean is pessimistic in the sense that we believe a controller to be unstable before seeing any data. The prior variance of k_{sim} and k_{err} are chosen as $\sigma^2_{\text{sim}} = 1.6 \times 10^{-5}$ and $\sigma^2_{\text{err}} = 3.84 \times 10^{-4}$ respectively.

The noise standard deviation of an evaluation on the real system, as defined in (2), has been estimated to $\eta_{\text{exp}} = 2.08 \times 10^{-4}$, while the noise of the simulator has been set to $\eta_{\text{sim}} = 10^{-5}$, roughly twenty times lower.

We stop the exploration when the GP posterior mean at the best guess $\boldsymbol{\theta}_{\text{bg}}$ (i.e., the current estimate of the global minimum) has not changed significantly (within a range of $\sigma_{\text{err}}/4$ over the last 3 iterations), and we are sufficiently certain about its value (posterior standard deviation at $\boldsymbol{\theta}_{\text{bg}}$ less than $\sigma_{\text{err}}/2$). Once the exploration has terminated, we evaluate the final best guess controller on a physical experiment and take its cost as the outcome of the learning procedure.

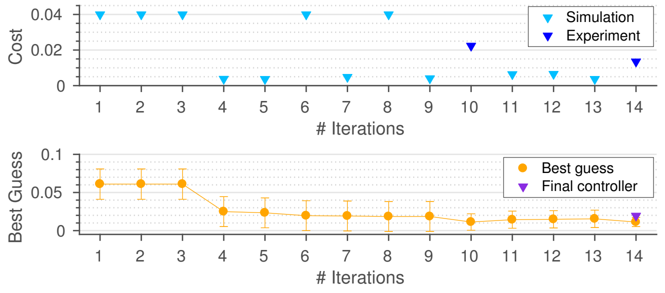


Fig. 4. (Top) Cost obtained at each iteration with the proposed approach during one exploration run. When the exploration terminates, the best guess is evaluated on the physical system (violet dot). (Bottom) Evolution of the GP posterior mean at the best guess $\mu_n(\boldsymbol{\theta}_{\text{bg}})$ and std $\pm\sigma_n(\boldsymbol{\theta}_{\text{bg}})$.

D. Results

We run MF-ES on the LQR problem described in Sec. V-B. Fig. 3 shows the final GP cost function landscape after the learning procedure, highlighting simulations (in light blue) and experiments (in dark blue). For the same learning run, Fig. 4 (top) illustrates how MF-ES alternates between simulations and physical experiments over iterations. As can be seen, the algorithm first performs multiple cheap simulations, which allow to identify regions of unstable controllers (i.e., regions of high predicted cost in Fig. 3) without any real experiment. At iterations 10 and 14, the algorithm demands two expensive physical experiments. The reason is that a time unit spent in simulation is expected to be less informative than on a physical experiment. Thereby, experiment time should be better spent on the physical system. Fig. 4 (bottom) shows the GP posterior mean and standard deviation of the best guess at each iteration. The stopping criterion terminates the exploration after 14 iterations because the GP posterior mean of the last three best guesses were steady enough. Finally, the algorithm selects the last global minimum, $\boldsymbol{\theta}_{\text{bg}} = [0.212,\ 2.42]$ (orange dot in Fig. 3), as the final controller, which was evaluated on the physical system retrieving a low cost $J(\boldsymbol{\theta}_{\text{bg}}) = 0.0194$.

As a remark, we observe that the algorithm alternates between simulations and experiments in a non-trivial way, which cannot be reproduced with a simple two-stage learning process, where simulations are used to seed experimental reinforcement learning. Furthermore, in Fig. 3, we can see that the posterior mean around $\boldsymbol{\theta} = [-2, 4]$ falls back to the prior in the absence of evaluations. As pointed out in Sec. V-C, the prior mean is pessimistic in the sense that predicts instability in unforeseen areas, which is a reasonable assumption in controller tuning of real systems.

In order to illustrate the benefit of trading off data from *experiments and simulations*, we compare MF-ES to ES [3], which uses *only physical experiments*. The latter corresponds to the automatic controller tuning setting in [14]. We run each of these methods ten times on the controller tuning problem. The results are discussed in Fig. 5 and Fig. 6.

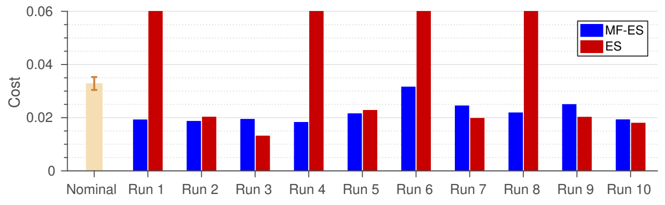


Fig. 5. Comparison of the final controller cost at each run between the proposed approach (MF-ES) and ES. The cost of the nominal controller (beige) with ± 2 std is shown for reference.

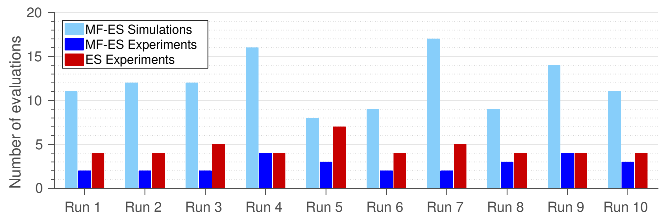


Fig. 6. Number of physical experiments at each run for MF-ES (dark blue) and standard ES (dark red), as well as simulations for MF-ES (light blue).

In Fig. 5, we show the cost of the final controller at each run, for both methods. The cost of the nominal controller (green) is shown as a reference. MF-ES finds controllers that are 33.23% better, on average. Moreover, it consistently finds stabilizing controllers, while ES fails to find a stabilizing solution in 4 out of 10 cases (cost of 0.06).

Fig. 6 compares the number of physical experiments performed with MF-ES (dark blue) and with ES (dark red). While ES needs on average 3.5 physical experiments, MF-ES needs 2.7 (22.86% less) plus 11.9 simulations. These results demonstrate that MF-ES can find, on average, better controllers with a lower number of real experiments by also leveraging information from simulations.

VI. Conclusion

We have shown a generic Bayesian optimization that can adaptively select between multiple information sources with different accuracies and evaluation efforts, such as experiments on a real robot and simulations. We applied this method to a policy optimization task on a cart-pole system. The experimental results confirm that using prior model information from a simulator can reduce the amount of data required to globally find good control policies.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 1998.

[2] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[3] P. Hennig and C. J. Schuler, “Entropy search for information-efficient global optimization,” *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 1809–1837, 2012.

[4] J. Villemonteix, E. Vazquez, and E. Walter, “An informational approach to the global optimization of expensive-to-evaluate functions,” *Journal of Global Optimization*, vol. 44, no. 4, pp. 509–540, 2008.

[5] M. Cutler and J. P. How, “Efficient reinforcement learning for robots using informative simulated priors,” in *IEEE International Conference on Robotics and Automation*, 2015, pp. 2605–2612.

[6] M. E. Taylor and P. Stone, “Transfer learning for reinforcement learning domains: A survey,” *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[7] M. Cutler, T. J. Walsh, and J. P. How, “Real-world reinforcement learning via multifidelity simulators,” *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 655–671, 2015.

[8] J. Peters and S. Schaal, “Policy gradient methods for robotics,” in *IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 2219–2225.

[9] P. Abbeel, M. Quigley, and A. Y. Ng, “Using Inaccurate Models in Reinforcement Learning,” in *ACM International Conference on Machine Learning*, 2006, pp. 1–8.

[10] J. Mockus, *Bayesian Approach to Global Optimization*, ser. Mathematics and Its Applications, M. Hazewinkel, Ed. Springer, 1989, vol. 37.

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

[12] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans, “Automatic gait optimization with Gaussian process regression.” in *International Joint Conference on Artificial Intelligence*, vol. 7, 2007, pp. 944–949.

[13] M. Tesch, J. Schneider, and H. Choset, “Using response surfaces and expected improvement to optimize snake robot gait parameters,” in *IEEE International Conference on Intelligent Robots and Systems*, 2011, pp. 1069–1074.

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

[15] H. Abdelrahman, F. Berkenkamp, J. Poland, and A. Krause, “Bayesian optimization for maximum power point tracking in photovoltaic power plants,” in *European Control Conference*, 2016, pp. 2078–2083.

[16] F. Berkenkamp, A. Krause, and Angela P. Schoellig, “Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics.” arXiv, 2016, arXiv:1602.04450 [cs.RO].

[17] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, “An experimental comparison of Bayesian optimization for bipedal locomotion,” in *IEEE International Conference on Robotics and Automation*, 2014, pp. 1951–1958.

[18] A. Krause and C. S. Ong, “Contextual Gaussian process bandit optimization,” in *Neural Information Processing Systems*, 2011, pp. 2447–2455.

[19] K. Swersky, J. Snoek, and R. P. Adams, “Multi-Task Bayesian Optimization,” in *Advances in Neural Information Processing Systems*, 2013, pp. 2004–2012.

[20] M. C. Kennedy and A. O’Hagan, “Predicting the output from a complex computer code when fast approximations are available,” *Biometrika*, vol. 87, no. 1, pp. 1–13, 2000.

[21] A. I. J. Forrester, A. Sóbester, and A. J. Keane, “Multi-fidelity optimization via surrogate modelling,” *Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007.

[22] K. Kandasamy, G. Dasarathy, J. B. Oliva, J. Schneider, and B. Poczos, “Multi-fidelity Gaussian process bandit optimisation.” arXiv, 2016, arXiv:1603.06288 [cs, stat].

[23] M. Poloczek, J. Wang, and P. I. Frazier, “Multi-Information Source Optimization with General Model Discrepancies.” arXiv, 2016, arXiv:1603.00389v2 [stat.ML].

[24] P. Frazier, W. Powell, and S. Dayanik, “The knowledge-gradient policy for correlated normal beliefs,” *INFORMS Journal on Computing*, vol. 21, no. 4, pp. 599–613, 2009.

[25] N. Srinivas, A. Krause, S. M. 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.

[26] Quanser, “Self-erecting single inverted pendulum – Instructor manual”, Tech. Rep. 516, rev. 4.1.

[27] S. Trimpe, A. Millane, S. Doessegger, and R. D’Andrea, “A self-tuning LQR approach demonstrated on an inverted pendulum,” in *19th IFAC World Congress*, 2014, pp. 11 281–11 287.

[28] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Mineola, New York: Dover Publications, 2007.

[29] J. Roberts, I. Manchester, and R. Tedrake, “Feedback controller parameterizations for reinforcement learning,” in *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning*, 2011, pp. 310–317.