Automatic LQR Tuning Based on Gaussian Process Optimization: Early Experimental Results

Alonso Marco[1], Philipp Hennig[2], Jeannette Bohg[1], Stefan Schaal[1,3] and Sebastian Trimpe[1]

***Abstract*— This paper proposes an automatic controller tuning framework based on linear optimal control combined with Bayesian optimization. With this framework, an initial set of controller gains is automatically improved according to a pre-defined performance objective evaluated from experimental data. The underlying Bayesian optimization algorithm is Entropy Search, which represents the latent objective as a Gaussian process and constructs an explicit belief over the location of the objective minimum. This is used to maximize the information gain from each experimental evaluation. Thus, this framework shall yield improved controllers with fewer evaluations compared to alternative approaches. A seven-degree-of-freedom robot arm balancing an inverted pole is used as the experimental demonstrator. Preliminary results of a low-dimensional tuning problem highlight the method's potential for automatic controller tuning on robotic platforms.**

I. INTRODUCTION

Robotic setups often need fine-tuned controller parameters both at low- and task-levels. Finding an appropriate set of parameters through simplistic protocols, such as manual tuning or grid search, can be highly time-consuming. We seek to automate the process of fine tuning a nominal controller based on performance observed in experiments on the physical plant. We aim for information-efficient approaches, where only few experiments are needed to obtain improved performance.

Designing controllers for balancing systems such as in [1] or [2] are typical examples for such a scenario. Often, one can without much effort obtain a rough linear model of the system dynamics around an equilibrium configuration, for example, from first principles modeling. Given the linear model, it is then relatively straightforward to compute a stabilizing controller, for instance, using optimal control. When testing this nominal controller on the physical plant, however, one may find the balancing performance unsatisfactory, e.g. due to unmodeled dynamics, parametric uncertainties of the linear model, sensor noise, or imprecise actuation. Thus, fine-tuning the controller gains in experiments on the real system is desirable in order to partly mitigate these effects and obtain improved balancing performance.

Aiming at automating this process, we propose a controller tuning framework extending previous work [3]. Therein, a Linear Quadratic Regulator (LQR) is iteratively improved based on control performance observed in experiments. The controller parameters of the LQR design are adjusted by means of a simple gradient descent approach, which performs local evaluations to compute a rough approximation of the gradient. While control performance could be improved in experiments on a balancing platform in [3], this approach does not exploit the available data as much as could be done. It uses the data only to locally approximate the gradient.

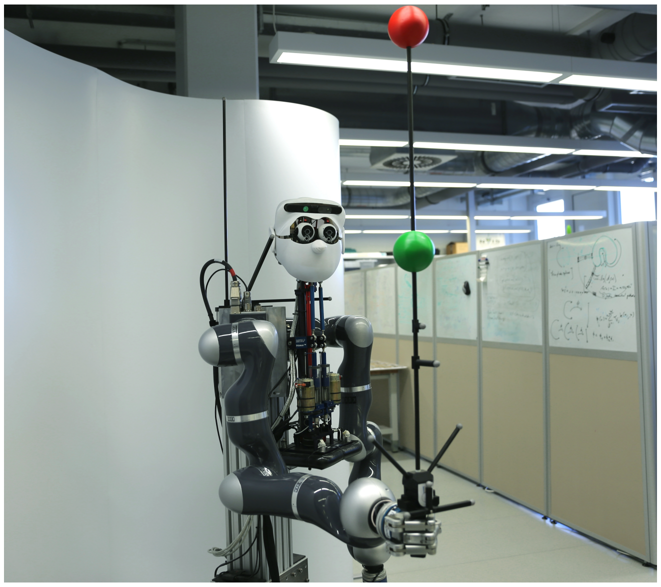
Fig. 1. Robot Apollo balancing an inverted pole. This experimental platform is used as a demonstrator of the automatic tuning framework.

In contrast to [3], we propose the use of Entropy Search (ES) [4], a recent algorithm for global Bayesian optimization, as the minimizer for the LQR tuning problem. ES employs a Gaussian process (GP) as a non-parametric model capturing the knowledge about the unknown cost function. At every iteration, the algorithm exploits all past data to infer the shape of the cost function. Furthermore, in the spirit of an active learning algorithm, it suggests the next evaluation such as to learn most about the cost function. Thus, we expect ES to be more data-efficient than simple gradient-based approaches as in [3]; that is, to yield better controllers with fewer experiments.

The main contribution of this paper is the combination of ES [4] with the LQR tuning framework proposed in [3]. The effectiveness of the resulting auto-tuning method is demonstrated in experiments of a humanoid robot balancing a pole

[1] Autonomous Motion Department at the Max Planck Institute for Intelligent Systems, Tübingen, Germany
[2] Empirical Inference Department at the Max Planck Institute for Intelligent Systems, Tübingen, Germany
[3] Computational Learning and Motor Control Lab at the University of Southern California, Los Angeles, CA, USA
E-mail: `alonso.marcovalle@tuebingen.mpg.de`
`strimpe@tuebingen.mpg.de`
This work was supported by the Max Planck Society.

in one dimension (see Figure 1). In this paper, we present first experimental results, where, in particular, we consider a low-dimensional parameterization of the controller. In order to test the learning method, we initialize it with a wrong model, but only modify one physical parameter (damping coefficient). Further experiments in higher dimensional spaces or with a completely wrong model are future work. Although ES has been successfully applied on numerical optimization problems before, this work is the first to use it for controller tuning on a complex robotic platform.

Related work: The LQR tuning framework herein considers the parametrization of controllers in terms of the weights of an LQR cost. In [5], this controller parametrization is explored in the context of reinforcement learning. The authors find this choice to be inefficient for solving a manipulation task. However, similar to the findings in [3], we find LQR parametrization suitable for improving feedback controllers for a balancing problem.

The cart-pole balancing problem is also used as a demonstrator in [6] and [7]. In contrast with our work, they parametrize directly the control feedback gain instead of the LQR weights, and the data is acquired from a simulated setup instead of a real system. In [6], the authors shape the instantaneous reward with a quadratic LQR cost, and retrieve the optimum after many roll-outs applying policy gradient. In [7], the parameters of a linear state feedback controller are learned. The space of parameters is explored by maximizing the posterior entropy of a GP that models the system performance through a cost function.

Outline of the paper: The LQR tuning problem is described in Sec. II. The use of ES for automating the tuning is outlined in Sec. III. The experimental results obtained on the robotic platform are presented in Sec. IV. The paper concludes with remarks in Sec. V.

II. LQR TUNING PROBLEM

In this section, we formulate the LQR tuning problem following the approach proposed in [3].

A. Control design problem

We consider a system that follows a discrete-time non-linear dynamic model

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) \tag{1}$$

with system states $\boldsymbol{x}_k$, control input $\boldsymbol{u}_k$, and zero-mean process noise $\boldsymbol{w}_k$ at time instant k. We assume that (1) has an equilibrium at $\boldsymbol{x}_k = \boldsymbol{0}$, $\boldsymbol{u}_k = \boldsymbol{0}$ and $\boldsymbol{w}_k = \boldsymbol{0}$, which we want to keep the system at. We also assume that $\boldsymbol{x}_k$ can be measured and, if not, an appropriate state estimator is used.

For regulation problems such as balancing about an equilibrium, a linear model is often sufficient for control design. Thus, we consider a scenario, where a linear model

$$\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{A}_{\text{n}}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}_{\text{n}}\boldsymbol{u}_k + \boldsymbol{w}_k \tag{2}$$

is given as an approximation of the dynamics (1) about the equilibrium at zero. We refer to (2) as the *nominal model*, while (1) are the true system dynamics, which are unknown.

A common way to measure the performance of a control system is through a quadratic cost function such as

$$J = \lim_{K\to\infty} \frac{1}{K}\mathbb{E}\left[\sum_{k=0}^{K} \boldsymbol{x}_k^T \boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R}\boldsymbol{u}_k\right] \tag{3}$$

with positive-definite weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, and $\mathbb{E}[\cdot]$ the expected value. The cost (3) captures a trade-off between control performance (keeping $\boldsymbol{x}_k$ small) and control effort (keeping $\boldsymbol{u}_k$ small), which we seek to achieve with the control design.

Ideally, we would like to obtain a state feedback controller for the non-linear plant (1) that minimized (3). Yet, this non-linear control design problem is intractable in general. Instead, a straightforward approach that yields a locally optimal solution is to compute the optimal controller minimizing (3) for the nominal model (2). This controller is given by the well-known Linear Quadratic Regulator (LQR) [8, Sec. 2.4]

$$\boldsymbol{u}_k = \boldsymbol{F}\boldsymbol{x}_k \tag{4}$$

whose static gain matrix $\boldsymbol{F}$ can readily be computed by solving the discrete-time infinite-horizon LQR problem for the nominal model $(\boldsymbol{A}_{\text{n}}, \boldsymbol{B}_{\text{n}})$ and the weights $(\boldsymbol{Q}, \boldsymbol{R})$. For simplicity, we write

$$\boldsymbol{F} = \text{lqr}(\boldsymbol{A}_{\text{n}}, \boldsymbol{B}_{\text{n}}, \boldsymbol{Q}, \boldsymbol{R}). \tag{5}$$

If (2) perfectly captured the true system dynamics (1), then (5) would be the optimal controller for the problem at hand. However, in practice, there can be several reasons why the controller (5) is suboptimal: the true dynamics are non-linear, the nominal linear model (2) involves parametric uncertainty, or the state is not perfectly measurable (e.g. noisy or incomplete state measurements). While still adhering to the controller structure (4), it is thus beneficial to fine tune the nominal design (the gain $\boldsymbol{F}$) based on experimental data to partly compensate for these effects. This is the goal of the automatic tuning approach, which is detailed next.

B. LQR tuning problem

Following the approach in [3], we parametrize the controller gains $\boldsymbol{F}$ in (4) as

$$\boldsymbol{F}(\boldsymbol{\theta}) = \text{lqr}(\boldsymbol{A}_{\text{n}}, \boldsymbol{B}_{\text{n}}, \bar{\boldsymbol{Q}}(\boldsymbol{\theta}), \bar{\boldsymbol{R}}(\boldsymbol{\theta})) \tag{6}$$

where $\bar{\boldsymbol{Q}}(\boldsymbol{\theta})$ and $\bar{\boldsymbol{R}}(\boldsymbol{\theta})$ are *design weights* parametrized in $\boldsymbol{\theta} \in \mathbb{R}^D$, which are to be varied in the automatic tuning procedure. For instance, $\bar{\boldsymbol{Q}}(\boldsymbol{\theta})$ and $\bar{\boldsymbol{R}}(\boldsymbol{\theta})$ can be diagonal matrices with $\theta_j > 0$, $j = 1, \ldots, D$, as diagonal entries.

When varying $\boldsymbol{\theta}$, different controller gains $\boldsymbol{F}(\boldsymbol{\theta})$ are obtained. These will affect the system performance through (4), thus resulting in a different cost value from (3) in each experiment. To make the parameter dependence of (3) explicit, we write

$$J = J(\boldsymbol{\theta}). \tag{7}$$

The goal of the automatic LQR tuning is to vary the parameters $\boldsymbol{\theta}$ such as to minimize the cost (3).

Remark: The weights $(\boldsymbol{Q}, \boldsymbol{R})$ in (3) are referred to as *performance weights*. Note that, while the *design weights*

$\big(\bar{\boldsymbol{Q}}(\boldsymbol{\theta}), \bar{\boldsymbol{R}}(\boldsymbol{\theta})\big)$ in (6) change during the tuning procedure, the performance weights remain unchanged.

C. Optimization problem

The above LQR tuning problem is summarized as the optimization problem

$$\arg\min J(\boldsymbol{\theta}) \quad \text{s.t. } \boldsymbol{\theta} \in \mathcal{D} \tag{8}$$

where we restrict the search of parameters to a bounded domain $\mathcal{D} \subset \mathbb{R}^D$. The domain $\mathcal{D}$ typically represents a region around the nominal design, where performance improvements are to be expected or exploration is considered to be safe.

The shape of the cost function in (8) is unknown. Neither gradient information is available nor guarantees of convexity can be expected. Furthermore, (3) cannot be computed from experimental data in practice as it represents an infinite-horizon problem. As is also done in [3], we thus consider the approximate cost

$$\hat{J} = \frac{1}{K}\left[\sum_{k=0}^{K} \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k\right] \tag{9}$$

with a finite, yet long enough horizon K. The cost (9) can be considered a noisy evaluation of (3). Such an evaluation is expensive as it involves conducting an experiment, which lasts few minutes in the considered balancing application.

III. LQR TUNING WITH ENTROPY SEARCH

In this section, we introduce Entropy Search (ES) [4] as the optimizer to address problem (8). The key characteristics of ES are explained in Sec. III-A to III-C, and the resulting framework for automatic LQR tuning is summarized in Sec. III-D. Here, we present only the high-level ideas of ES from a practical standpoint. The reader interested in the mathematical details, as well as further explanations, is referred to [4].

A. Underlying cost function as a Gaussian process

ES is one out of several popular formulations of Bayesian Optimization [9], [10], [11], a framework for global optimization in which uncertainty over the objective function J is represented by a probability measure $p(J)$, typically a Gaussian process (GP) [12]. Note that the shape of the cost function (3) is unknown; only noisy evaluations (9) are available. A GP can be understood as a probability measure over a function space. Thus, the GP encodes the knowledge that we have about the cost function. New evaluations are incorporated through conditioning the GP on these data. With more data points, the cost function shape thus becomes better known. GP regression is a common way in machine learning for inferring an unknown function from noisy evaluations; refer to [12] for more details.

We model prior knowledge about the cost function J as the GP

$$J(\boldsymbol{\theta}) \sim \mathcal{GP}\left(\mu(\boldsymbol{\theta}), k(\boldsymbol{\theta}, \boldsymbol{\theta}_*)\right) \tag{10}$$

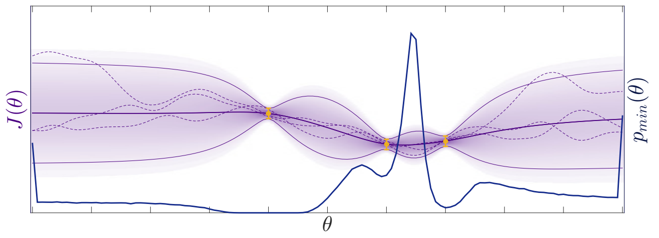

Fig. 2. Example Gaussian process after three function evaluations (orange dots), reproduced with slight alterations from [4]. The posterior mean $\bar{\mu}(\boldsymbol{\theta})$ is shown in solid thick violet, two standard deviations $2\bar{\sigma}(\boldsymbol{\theta})$ in solid thin violet, and the probability density as a gradient of color that decreases away from the mean. Two standard deviations of the likelihood noise $2\sigma_\text{n}$ are represented as orange vertical bars at each evaluation. Three randomly sampled functions from the posterior GP as dashed violet lines. Approximated probability distribution over the location of the minimum $p_\text{min}(\boldsymbol{\theta})$ in dark blue. This plot uses arbitrary scales for each object.

with mean function $\mu(\boldsymbol{\theta})$ and covariance function $k(\boldsymbol{\theta}, \boldsymbol{\theta}_*)$. Common choices are a zero mean function ($\mu(\boldsymbol{\theta}) = \boldsymbol{0}$ for all $\boldsymbol{\theta}$), and the squared exponential (SE) covariance function

$$k_\text{SE}(\boldsymbol{\theta}, \boldsymbol{\theta}_*) = \sigma^2 \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\text{T} \boldsymbol{S} (\boldsymbol{\theta} - \boldsymbol{\theta}_*)\right] \tag{11}$$

which we also use herein. The covariance function $k(\boldsymbol{\theta}, \boldsymbol{\theta}_*)$ generally measures covariance between $J(\boldsymbol{\theta})$ and $J(\boldsymbol{\theta}_*)$. It can thus be used to encode assumptions about properties of J such as smoothness, characteristic length-scales, and signal variance. In particular, the SE covariance function (11) models smooth functions with signal variance σ^2 and length-scales $\boldsymbol{S} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$, $\lambda_j > 0$.

We assume that the noisy evaluations (9) of (3) can be modeled as

$$\hat{J} = J(\boldsymbol{\theta}) + \varepsilon \tag{12}$$

where the independent identically distributed noise ε describes a Gaussian likelihood $\mathcal{N}(J(\boldsymbol{\theta}), \sigma_\text{n}^2)$.

To simplify notation, we write $\boldsymbol{y} = \{\hat{J}^i\}_{i=1}^N$ for N evaluations at locations $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^i\}_{i=1}^N$. Conditioning the GP on the data $\{\boldsymbol{y}, \boldsymbol{\Theta}\}$ then yields another GP with posterior mean $\bar{\mu}(\boldsymbol{\theta})$ and a posterior variance $\bar{k}(\boldsymbol{\theta}, \boldsymbol{\theta}_*)$.

Figure 2 provides an example for a one-dimensional cost function. Shown are the posterior mean and two standard deviations of the GP after three evaluations (orange dots). As can be seen from this graph, the shape of the mean is adjusted to fit the data points, and the uncertainty (standard deviation) is reduced around the evaluations points. In regions where no evaluations have been made, the uncertainty is still large. Thus, the GP provides a mean approximation of the underlying cost function, as well as a description of the uncertainty associated with this approximation.

We gather the hyperparameters of the GP in the set $\mathcal{H} = \{\lambda_1, \lambda_2, \ldots, \lambda_D, \sigma, \sigma_n\}$. An initial choice of $\mathcal{H}$ can be improved with every new data point $\hat{J}^i$ by adapting the hyperparameters. As commonly done, the marginal likelihood is maximized with respect to $\mathcal{H}$ at each iteration of ES.

Typically, the cost can have different sensitivity to different parameters θ_j. We use automatic relevance determination

[12, Sec. 5.1] in the covariance function (11) to remove from the inference those dimensions that have low impact on the cost.

B. Probability measure over the location of the minimum

A key idea of ES (see [4, Sec. 1.1]) is to explicitly represent the probability $p_{\min}(\boldsymbol{\theta})$ for the minimum location:

$$p_{\min}(\boldsymbol{\theta}) \equiv p(\boldsymbol{\theta} = \arg\min J(\boldsymbol{\theta})). \tag{13}$$

The probability $p_{\min}(\boldsymbol{\theta})$ is induced by the GP for J: given a distribution of cost functions J as described by the GP, one can in principle compute the probability for any $\boldsymbol{\theta}$ of being the minimum of J. For the example GP in Fig. 2, $p_{\min}(\boldsymbol{\theta})$ is shown by the dark blue line.

To obtain a tractable algorithm, ES approximates $p_{\min}(\boldsymbol{\theta})$ with finitely many points on a non-uniform grid that puts higher resolution in regions of greater influence.

C. Information-efficient evaluation decision

The key feature of ES is the suggestion of new locations $\boldsymbol{\theta}$, where (9) should be evaluated to learn most about the location of the minimum. This is achieved by selecting the next evaluation point that maximizes the relative entropy

$$H = \int_{\mathcal{D}} p_{\min}(\boldsymbol{\theta}) \log \frac{p_{\min}(\boldsymbol{\theta})}{b(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} \tag{14}$$

between $p_{\min}(\boldsymbol{\theta})$ and the uniform distribution $b(\boldsymbol{\theta})$ over the bounded domain $\mathcal{D}$. The rationale for this is that the uniform distribution essentially has no information about the location of the minimum, while a very "peaked" distribution would be desirable to obtain distinct potential minima. This can be achieved by maximization of the relative entropy (14). The associated problem is solved numerically.

ES selects next evaluations where the first order expansion $\Delta H(\boldsymbol{\theta})$ of the expected change in (14) is maximal. In this way, the algorithm efficiently explores the domain of the optimization problem in terms of information gain (cf. [4, Sec. 2.5]). Conceptually, the choice of the locations $\boldsymbol{\Theta}$ is made such that "we evaluate where we expect to learn most about the minimum, rather than where we think the minimum is" [4, Sec. 1.1].

In addition to suggesting the next evaluation, ES also outputs its current best guess of the minimum location; that is, the maximum of its approximation to $p_{\min}(\boldsymbol{\theta})$.

D. Automatic LQR tuning

The proposed method for automatic LQR tuning is obtained by combining the LQR tuning framework from Section II with ES; that is, using ES to solve (8). At every iteration, ES suggests a new controller (through $\boldsymbol{\theta}$ with (6)), which is then tested in an experiment to obtain a new cost evaluation (9). Through this iterative procedure, the framework is expected to explore relevant regions of the cost (3), infer the shape of the cost function, and eventually yield the global minimum within $\mathcal{D}$. The automatic LQR tuning method is summarized in Algorithm 1.

The performance weights $(\boldsymbol{Q}, \boldsymbol{R})$ encode the desired performance for the system (1). Thus, a reasonable initial choice of the parameters $\boldsymbol{\theta}$ is such that the design weights $(\bar{\boldsymbol{Q}}(\boldsymbol{\theta}), \bar{\boldsymbol{R}}(\boldsymbol{\theta}))$ equal $(\boldsymbol{Q}, \boldsymbol{R})$. The obtained initial gain $\boldsymbol{F}$ would be optimal if (2) were the true dynamics. After N evaluations, ES is expected to improve this initial gain based on experimental data representing the true dynamics (1).

The call to ES involves specifying the type of covariance function k and the type of likelihood l, while it assumes zero mean.

Algorithm 1 Automatic LQR Tuning.

```
 1: initialize $\boldsymbol{\theta}^0$; typically $\bar{\boldsymbol{Q}}(\boldsymbol{\theta}^0) = \boldsymbol{Q}$, $\bar{\boldsymbol{R}}(\boldsymbol{\theta}^0) = \boldsymbol{R}$
 2: $\hat{J}^0 \leftarrow$ CostEvaluation($\boldsymbol{\theta}^0$)                ▷ Cost evaluation
 3: $\{\boldsymbol{\Theta}, \boldsymbol{y}\} \leftarrow \{\boldsymbol{\theta}^0, \hat{J}^0\}$
 4: procedure EntropySearch($k$,$l$,$N$,$\{\boldsymbol{\Theta}, \boldsymbol{y}\}$)
 5:     for $i = 1$ to $N$ do
 6:         $[\bar{\mu}, \bar{k}] \leftarrow$ GP($k, l, \{\boldsymbol{\Theta}, \boldsymbol{y}\}$)         ▷ GP posterior
 7:         $p_{\min} \leftarrow$ approx($\bar{\mu}, \bar{k}$)              ▷ Approximate $p_{\min}$
 8:         $\boldsymbol{\theta}^i \leftarrow \arg\max \Delta H$          ▷ Next location to evaluate at
 9:         $\hat{J}^i \leftarrow$ CostEvaluation($\boldsymbol{\theta}^i$)           ▷ Cost evaluation
10:         $\{\boldsymbol{\Theta}, \boldsymbol{y}\} \leftarrow \{\boldsymbol{\Theta}, \boldsymbol{y}\} \cup \{\boldsymbol{\theta}^i, \hat{J}^i\}$
11:         $\boldsymbol{\theta}^{\mathrm{BG}} \leftarrow \arg\max p_{\min}$       ▷ Update current "Best Guess"
12:     end for
13:     return $\boldsymbol{\theta}^{\mathrm{BG}}$
14: end procedure

15: function CostEvaluation($\boldsymbol{\theta}$)
16:     LQR design: $\bar{\boldsymbol{F}} \leftarrow$ lqr($\boldsymbol{A}_{\mathrm{n}}, \boldsymbol{B}_{\mathrm{n}}, \bar{\boldsymbol{Q}}(\boldsymbol{\theta}), \bar{\boldsymbol{R}}(\boldsymbol{\theta})$)
17:     update control law (4) with $\boldsymbol{F} = \bar{\boldsymbol{F}}$
18:     perform experiment and record $\{\boldsymbol{x}_k\}, \{\boldsymbol{u}_k\}$
19:     Evaluate cost: $\hat{J} \leftarrow \frac{1}{K}\left[\sum_{k=0}^{K} \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k\right]$
20:     return $\hat{J}$
21: end function
```

IV. Experimental Results

Starting with the initial LQR design (5) based on the nominal model and the performance weights, ES adjusts the design (6) and guides the search towards a controller that retrieves a better performance, and thus, a lower cost value.

This paper shows preliminary results of the proposed framework in a two-dimensional parameter space.

A. System description

We consider a one-dimensional balancing problem: a pole linked to a handle through a rotatory joint with one degree of freedom (DOF) is kept upright by controlling the acceleration of the end-effector of a seven DOF robot arm (Kuka lightweight robot). Figure 1 shows the setup with the pole at the upright position. The angle of the pole is tracked using an external motion capture system. The two colored balls do not serve any functional purpose in this project.

The continuous-time dynamics of the balancing problem (similar to [13]) are described by:

$$\begin{aligned} &mr^2\ddot{\psi}(t) - mgr\sin\psi(t) + mr\cos\psi(t)u(t) + \xi\dot{\psi}(t) = 0 \\ &\ddot{s}(t) = u(t) \end{aligned} \tag{15}$$

where $\psi(t)$ is the deviation of the pole angle with respect to the gravity axis, $s(t)$ is the deviation of the end-effector from the zero position, and $u(t)$ represents the acceleration of the the end-effector. The center of mass of the pole lies at $r \simeq 0.61$ m from the axis of the rotatory joint, its mass

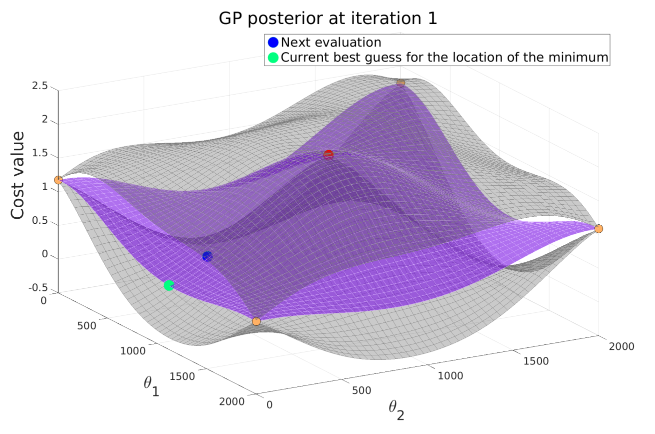

Fig. 3. Mean of the GP represented in violet and two standard deviations (above and below) in grey. Current “best guess” for the location of the minimum (green dot) $\boldsymbol{\theta}^{\mathrm{BG}} = [1118.3, 1.0]$. Location suggested to evaluate next such that the information gain is maximal (blue dot) $\boldsymbol{\theta}^{6} = [1086.01, 245.4]$. The red dot corresponds to the initial controller, computed at location $\boldsymbol{\theta}^{0} = \left[10^3, 10^3\right]$. In general, the orange dots represent any other evaluation. In this case, these occur at the corners of the domain.

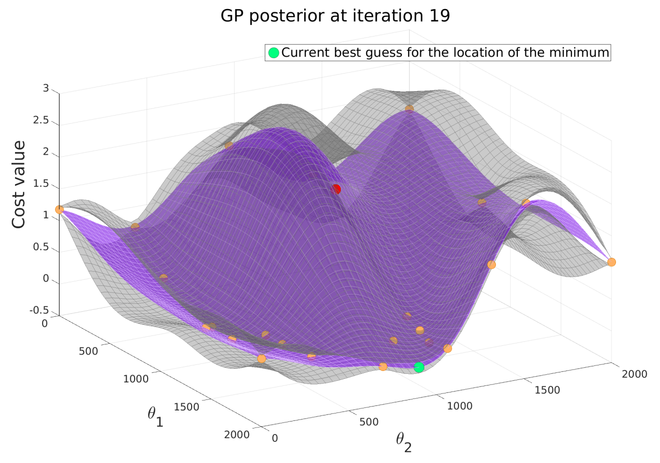

Fig. 4. GP posterior after 24 evaluations. After 19 iterations ES suggest $\boldsymbol{\theta}_{BG} = [1999.9, 899.4]$ as the location of the minimum (green dot). The red dot corresponds to the initial controller.

is $m \simeq 0.39$ kg, the ball bearing in the rotatory joint has a friction coefficient of $\xi \simeq 0.012$ Nms, and the gravity constant is $g = 9.81$ m/s^2.

The model (15) assumes that we can command a discretized end-effector acceleration u_k as control input to the system. In reality, this end-effector acceleration is realized through an appropriate tracking controller for the end-effector. For this tracking controller, we follow a similar control structure as the one proposed in [14].

A model (2) of the system is obtained by linearization of (15) about the equilibrium $\psi = 0$, $s = 0$ and discretization with a sampling time of $T_s = 1$ ms.

The estimated end-effector position s_k and velocity $\dot{s}_k$ are computed at a sampling rate of 1kHz from the robot's joint encoders using forward kinematics. The pole orientation is captured at 200 Hz by the motion capture system; this measurement is fed to a Kalman filter estimating the pole angle ψ_k and its angular velocity $\dot{\psi}_k$.

We augment the system by an integrator on s_k, $z_{k+1} = z_k + T_{\mathrm{s}} s_k$ to compensate for steady state error in the end-effector position (see [1] for an analysis of integral action in balancing problems). With this, the entire state vector is

$$\boldsymbol{x}_k = [\psi_k, \dot{\psi}_k, s_k, \dot{s}_k, z_k]^{\mathrm{T}}. \tag{16}$$

B. Automatic LQR tuning: Implementation choices

We choose the performance weights to be

$$\boldsymbol{Q} = \mathrm{diag}(1, 10^3, 1, 10^3, 0), \ \boldsymbol{R} = 1 \tag{17}$$

where diag$(\cdot)$ denotes the diagonal matrix with the arguments on the diagonal. We desire to have a quiet overall motion in the system. Therefore, we penalize the velocities $\dot{\psi}_k$ and $\dot{s}_k$ more than the other states. The weight for the integrator state is set to zero, as it is an artificial state (implemented in the controller), which does not affect the perceived balancing performance, but only the steady-state position of the end-effector.

We parametrize the design weights as

$$\bar{\boldsymbol{Q}}(\boldsymbol{\theta}) = \mathrm{diag}(1, \theta_1, 1, \theta_2, 1), \ \bar{\boldsymbol{R}}(\boldsymbol{\theta}) = 1 \tag{18}$$

where the parameters $\boldsymbol{\theta} = [\theta_1, \theta_2]$ are allowed to vary in the range $[1, 2000]$.

An initial choice for the design weights is such that $\boldsymbol{\theta}^0 = \left[10^3, 10^3\right]$, since we want the first four states to have the same penalization as the one required in the performance weights. In favour of having a feasible LQR design, a non-zero weight must be chosen for the integrator state. Although this makes (18) differ from (17), we still expect ES to improve any arbitrarily chosen initial controller.

Balancing experiments were run for 2 minutes, i.e. a discrete time horizon of $K = 1.2 \cdot 10^5$ steps. However, some controllers could destabilize the system at time $m < K$, which means that the system exceeded either acceleration bounds or state constraints (in particular, a box of allowable end-effector positions) introduced for safety reasons. In these cases, the experiment was stopped and the cost was computed using the available data up to the time m using the heuristic

$$\hat{J}_{\mathrm{uns}} = \frac{1}{K}\left[\sum_{k=0}^{m-1}\left(\boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k\right) + \alpha(K - m)\right] \tag{19}$$

where $\alpha > 0$ was chosen roughly one order of magnitude larger than the typical cost values obtained under stable conditions.

Before running ES, a few experiments were realized to acquire knowledge about the hyperparameters $\mathcal{H}$. Accordingly, a Gamma prior distribution was assumed over each hyperparameter with expected values $\mathbb{E}[\lambda_j] = 500$, $\mathbb{E}[\sigma] = 0.4$, $\mathbb{E}[\sigma_{\mathrm{n}}] = 0.01$, which were used as prior knowledge for the first iteration of ES, assuming a small variance. After each iteration, the GP marginal likelihood was maximized with respect to the hyperparameters taking into account these Gamma priors to better fit the current data.

TABLE I

COST VALUES FOR CASES 1 AND 2

		$\hat{J}$ (Case 1)		$\hat{J}$ (Case 2)	
		mean	std	mean	std
Initial set	$\boldsymbol{\theta}^0$	1.862	-	0.1958	0.02800
Final best guess	$\boldsymbol{\theta}^{\mathrm{BG}}$	0.17556	0.0208	0.1648	0.02162

C. Results

We present the experimental results of two runs of the automatic LQR tuning. In the first one, the model (2) has been corrupted (significant underestimation of the damping coefficient) to provide more room for the auto-tuning. In the second one, we used the best available linear model.

Case 1: ES was initialized with five evaluations, i.e. the initial controller $\boldsymbol{\theta}^0$, and evaluations at the four corners of the domain $[1, 2000] \times [1, 2000]$. The initial controller destabilized the system and its cost was computed according to (19) with $\alpha = 1.5$. Figure 3 shows the two-dimensional Gaussian process posterior after the five initial evaluations, which have relatively high cost values.

The algorithm can also work without evaluating initially at the corners of the domain; however, we found that they provide useful prestructuring of the GP and tend to speed up the learning. This way, the algorithm focuses on interesting regions more quickly.

Executing Algorithm 1 for 19 iterations (i.e. 19 balancing experiments) yielded the posterior GP illustrated in Fig. 4. The "best guess" $\boldsymbol{\theta}_{BG}$ (green dot) is what ES suggests to be the location of the minimum of the underlying cost (3).

To evaluate the result of the automatic LQR tuning, we computed the cost of the resulting controller (best guess after 19 iterations) in five separate balancing experiments. The average and standard deviation of these experiments are shown in Table I (left), together with the cost value of the initial controller (which was unstable in this case).

Case 2: In this experiment, the parameters of the model were not artificially modified, and the initial controller was stable. After running Algorithm 1 for 27 iterations, ES suggested $\boldsymbol{\theta}_{BG} = [1402.1, 451, 6]$ as the best controller. The results of evaluating this controller five times, in comparison to the initial controller, are shown in Table I (right). Albeit the initial controller was obtained from the best linear model we had, the performance could still be improved by 15.9%.

V. CONCLUDING REMARKS

In this work, we introduce Entropy Search (ES), a global Bayesian optimization algorithm, for automatic controller tuning. We successfully demonstrated the developed tuning framework in experiments of a humanoid robot balancing an inverted pendulum. Improved controllers were obtained both when the method was initialized with an unstable and with a stable controller.

The experiments herein represent early results, and we plan to do further tests in order to better evaluate the potential of the proposed method. In particular, we plan to investigate higher dimensional tuning problems (here, only two parameters were varied) and cases where only a completely wrong nominal model is given (here, only the damping coefficient was corrupted). Since the ES algorithm reasons about where to evaluate next in order to maximize the information gain of an experiment, we expect the algorithm to make better use of the available data and yield improved controllers more quickly than alternative approaches. Investigating whether this claim holds in practice by comparing the tuning performance of different methods is future work.

Another relevant direction for future extensions of the proposed method toward a truly automatic tool concerns the aspect of safety. While ES seeks to maximize the information obtained from an experiment, it seems reasonable to also include safety considerations such as avoiding unstable controllers. Safe learning in the context of control is an area of increasing interest and has recently been considered, e.g., in [7], [15], [16] and [17] in slightly different contexts.

REFERENCES

[1] S. Trimpe and R. D'Andrea, "The Balancing Cube: A dynamic sculpture as test bed for distributed estimation and control," *IEEE Control Systems Magazine*, vol. 32, no. 6, pp. 48–75, 2012.

[2] S. Mason, L. Righetti, and S. Schaal, "Full dynamics LQR control of a humanoid robot: An experimental study on balancing and squatting," in *14th IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 374–379.

[3] S. Trimpe, A. Millane, S. Doessegger, and R. D'Andrea, "A self-tuning LQR approach demonstrated on an inverted pendulum," in *IFAC World Congress*, 2014, pp. 11 281–11 287.

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

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

[6] J. Peters and S. Schaal, "Natural Actor-Critic," *Neurocomputing*, vol. 71, no. 7, pp. 1180–1190, 2008.

[7] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint, "Safe exploration for active learning with Gaussian processes," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2015, to appear.

[8] B. D. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Dover Publications, 1990.

[9] 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.

[10] D. J. Lizotte, "Practical bayesian optimization," Ph.D. dissertation, 2008.

[11] M. A. Osborne, R. Garnett, and S. J. Roberts, "Gaussian processes for global optimization," in *3rd International Conference on Learning and Intelligent Optimization*, 2009, pp. 1–15.

[12] C. E. Rasmussen, "Gaussian processes for machine learning," 2006.

[13] S. Schaal, "Learning from demonstration," *Advances in Neural Information Processing Systems*, pp. 1040–1046, 1997.

[14] L. Righetti, M. Kalakrishnan, P. Pastor, J. Binney, J. Kelly, R. C. Voorhies, G. S. Sukhatme, and S. Schaal, "An autonomous manipulation system based on force control and optimization," *Autonomous Robots*, vol. 36, no. 1-2, pp. 11–30, 2014.

[15] A. K. Akametalu, S. Kaynama, J. F. Fisac, M. N. Zeilinger, J. H. Gillula, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *IEEE 53rd Annual Conference on Decision and Control*, 2014, pp. 1424–1431.

[16] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with Gaussian processes," in *The 32nd International Conference on Machine Learning*, 2015, pp. 997–1005.

[17] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *European Control Conference*, 2015.