Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

A Self-Tuning LQR Approach Demonstrated on an Inverted Pendulum

Sebastian Trimpe * **Alexander Millane** ** **Simon Doessegger** **
Raffaello D'Andrea **

* *Max Planck Institute for Intelligent Systems, Autonomous Motion Department, Tübingen, Germany (e-mail: strimpe@tuebingen.mpg.de).*
** *Institute for Dynamic Systems and Control, ETH Zurich, Switzerland (millanea@student.ethz.ch, doessegger.s@gmail.com, rdandrea@ethz.ch).*

Abstract: An automatic controller tuning approach is presented that iteratively updates a linear quadratic regulator (LQR) design such that the resulting controller achieves improved closed-loop performance. In each iteration, an updated LQR gain is obtained by adjusting the weighting matrices of the associated quadratic cost. The performance of the resulting controller (measured in terms of another quadratic cost with fixed weights) is evaluated from experimental data obtained by testing the controller in closed-loop operation. The weight adjustment occurs through a stochastic optimization seeking to minimize the experimental cost. Simulation results of a stochastic linear system show that the self-tuning algorithm can recover optimal performance despite having imprecise model knowledge. Experiments on an inverted pendulum demonstrate that the method is effective in improving the system's balancing performance.

Keywords: linear quadratic control, adaptive control, automatic tuning, balancing systems.

1. INTRODUCTION

The primary purpose of sensor measurements in any feedback control system is to provide information about the state of the dynamic system under control for making appropriate control decisions. However, sensor data typically also provides information about the current operating conditions (such as performance) of the feedback control system as a whole. Inverted pendulum systems that are actively kept in balance by a feedback control system provide a good example: firstly, sensory information about the pendulum's state is required for stabilizing the pendulum about its unstable equilibrium. Secondly, the sensor data reveals how well the control system is doing: large variations of the pendulum state about the equilibrium indicate poor balancing performance. In this paper, we present a self-tuning algorithm that exploits this secondary use of sensor data by automatically adjusting feedback gains based on observed performance in experiments.

This work is motivated by the Balancing Cube (see Trimpe and D'Andrea [2012]), which is a cubic structure being balanced on one of its corners by six rotating arms, and essentially represents a 3D multi-body inverted pendulum. When the cube is balancing, it is not at a perfect standstill, but exhibits slight motion caused by, for example, sensor noise or imperfections in the actuation mechanism.

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

$$J = \lim_{K\to\infty} \frac{1}{K}\,\mathbb{E}\left[\sum\nolimits_{k=0}^{K-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k\right], \tag{1}$$

with states x_k, control inputs u_k, weighting matrices Q and R, and $\mathbb{E}[\cdot]$ the expected value. For a linear system with a *known model* and *perfect state measurements*, the controller that minimizes (1) is the well-known Linear Quadratic Regulator (LQR), which is a static feedback controller

$$u_k = F x_k \tag{2}$$

whose gain matrix F can efficiently be computed from the system model and the weighting matrices (see Anderson and Moore [2007], for example). That is, in this ideal situation, no controller tuning is required.

In practice, however, a system model is typically only known approximately, and when the gain F is computed from this approximate model, (2) is no longer the optimal controller. In this situation, it is desirable to adjust the controller gain to recover the optimal performance.

The calculation of the LQR gain F depends on (i) the system model and (ii) the choice of weighting matrices. Therefore, one has, in principle, two ways of adjusting the LQR gains: (i) by improving the system model, or (ii) by modifying the weighting matrices. Approach (i) is in line with indirect adaptive control, where the adjustment process commonly occurs in two stages: first, model parameters are identified from sensor data and, second, the updated model is used to determine a controller, for example, by solving the LQR problem (see Åström and Wittenmark [2008], Grimble [1984], Clarke et al. [1985], for example). In contrast, we pursue a direct approach according to (ii), where we modify the weighting matrices directly while leaving the system model unchanged. This way, a separate system identification step is avoided.

The basic mechanism of the self-tuning LQR approach is as follows. We introduce a separate set of weighting matrices $\bar{Q}$ and $\bar{R}$, which we use, together with an approximate system model, to compute LQR gains. The performance of an LQR controller is evaluated by using it in closed-loop operation on the physical plant and computing the cost (1) from experimental data. A superordinate optimization algorithm uses this information to iteratively update $\bar{Q}$ and $\bar{R}$ in order to improve the LQR design.

Even though we motivate the self-tuning LQR approach for the case of a linear process with imprecisely known parameters, the method may be regarded as a general approach for automatically tuning static feedback controllers. Essentially, the method can be used whenever tuning of a controller (2) based on experimental data is desirable. In addition to the scenario of an imprecise model, we also discuss the method's effectiveness in obtaining improved static feedback controllers for the case of imperfect (noisy) state measurements.

Directly tuning the weights of an LQR cost to improve control performance is also considered by Roberts et al. [2011], who discuss different parameterizations of feedback controllers in the context of reinforcement learning (one parameterization being through LQR weights). The authors consider a full parameterization of the LQR weights, whereas we introduce suitable parameterizations of the weights to reduce the dimensionality of the learning/tuning problem. While Roberts et al. [2011] find parameterization through LQR weights ineffective for improving control performance in their application, we show in this paper that automatically tuning LQR weights can be effective indeed: we present a simulation example where perfect compensation for an imprecise model is achieved and experiments on an inverted pendulum where balancing performance improves by about 20% (compared to the controller designed under the idealizing assumptions of perfect model knowledge and perfect state measurements).

After introducing notation and summarizing the standard LQR in the next section as a basis for the development, the self-tuning LQR algorithm is formalized in Sec. 3. By means of simulation examples in Sec. 4, we demonstrate the algorithm's effectiveness in improving controller performance for the case of imperfect model knowledge. Section 5 then presents experimental results of testing the method on an inverted pendulum system, which represents a 1D abstraction of the Balancing Cube. The paper concludes with remarks in Sec. 6.

2. NOTATION AND PRELIMINARIES

$\mathrm{Var}[x]$ denotes the variance of a vector-valued random variable x. Where convenient, vectors are expressed as tuples $(v_1, v_2, \dots)$ with dimension and stacking clear from context. For the data sequence $x_0, x_1, \dots, x_K$, we write $\{x_k\}_{k=0}^{K}$, or $\{x_k\}$ if the horizon K is clear from context.

For a symmetric matrix $X \in \mathbb{R}^{n \times n}$, we write $X > 0$ and $X \geq 0$ to mean that X is positive definite and positive semi-definite, respectively. The trace of a matrix is denoted by $\mathrm{tr}(\cdot)$, and $\mathrm{diag}(x_1, x_2, \dots)$ denotes the diagonal matrix with $x_1, x_2, \dots$ on the diagonal.

Linear Quadratic Regulator (LQR)

Consider the stochastic, linear time-invariant (LTI) system

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{3}$$

the static state-feedback controller (2), and the infinite horizon quadratic cost (1); where $x_k \in \mathbb{R}^n$ is the state; $u_k \in \mathbb{R}^m$ is the control input; $w_k \in \mathbb{R}^n$ is independent identically distributed (i.i.d.) process noise with $\mathbb{E}[w_k] = 0$ and $\mathrm{Var}[w_k] = W_\mathrm{n}$; A, B and F are real matrices of appropriate dimensions; $Q \geq 0$ and $R > 0$ are symmetric weight matrices; and $k \geq 0$ is the discrete-time index. We generally assume that (A, B) is stabilizable and (A, D) with $Q = D^\mathrm{T} D$ is detectable.

For given problem parameters A, B, W_n, Q, and R, the cost (1) depends on the choice of feedback gain F; hence, we often write $J(F)$ for (1). We denote the quadratic cost computed over a finite horizon of length K (for example, from experimental data $\{x_k\}$ and $\{u_k\}$) by $\hat{J} = \hat{J}(F)$; that is,

$$\hat{J} := \frac{1}{K} \sum_{k=0}^{K-1} x_k^\mathrm{T} Q x_k + u_k^\mathrm{T} R u_k \tag{4}$$

and $J = \lim_{K \to \infty} \mathbb{E}[\hat{J}]$. If the process (3) is stationary and the horizon K is long enough, then $\hat{J} \approx J$.

The controller that minimizes (1) for the system (3) is the Linear Quadratic Regulator (LQR) given by (2) and a specific gain $F = F^*$ that can readily be computed from the problem parameters A, B, Q, and R using standard tools (see Anderson and Moore [2007], for example). For simplicity, we omit the details of its computation and write

$$F^* = \mathrm{lqr}(A, B, Q, R)\,. \tag{5}$$

3. SELF-TUNING LQR

In this section, we introduce the controller design problem and develop the self-tuning LQR approach thereafter.

3.1 Problem Statement

The overall objective is to design a controller for the stochastic LTI process (3) such that the quadratic cost (1) is minimized for the situation where the system model (A, B) is not known, but only an approximate model

$$\bar{A} \approx A, \quad \bar{B} \approx B \tag{6}$$

is available for controller design. We refer to (A, B) as the *true system*, and to $(\bar{A}, \bar{B})$ as the *nominal model*. We first assume that we have perfect state measurements

$$y_k = x_k. \tag{7}$$

Later, in Sec. 3.4, we address the case of noisy state measurements. We assume that we have access to data $\{y_k\}$, $\{u_k\}$ generated by the true system (3), for example, from experiments.

The controller that minimizes (1) is the LQR presented in Sec. 2; that is, the static feedback controller

$$u_k = Fy_k \tag{8}$$

with $F = F^*$ as in (5). If the true system (A, B) was known, we could simply compute this controller. The LQR gain that is computed with the approximate model $(\bar{A}, \bar{B})$,

$$F = \bar{F}^0 := \mathrm{lqr}(\bar{A}, \bar{B}, Q, R), \tag{9}$$

is, however, not optimal. Though often, the controller (2) with gain (9) will yield reasonable performance provided that the discrepancy between (A, B) and $(\bar{A}, \bar{B})$ is not too large. Therefore, (9) serves as the *initial design* in the proposed method.

We are interested in developing an automatic tuning method that systematically improves the initial design (9) by exploiting experimental data $\{y_k\}$, $\{u_k\}$ obtained during closed-loop operation of the system. Ideally, we wish to recover the optimal cost $J^* = J(F^*)$ this way, despite having an inaccurate system model.

3.2 Self-Tuning Through Variation of LQR Weights

In contrast to system identification-based approaches, we leave the nominal model $(\bar{A}, \bar{B})$ unchanged, but modify the weights used in the LQR design directly in order to obtain controllers that are superior to the initial design (9). For this purpose, we introduce the *design weights* $\bar{Q}$ and $\bar{R}$ and compute controller gains as

$$F = \bar{F} := \mathrm{lqr}(\bar{A}, \bar{B}, \bar{Q}, \bar{R}). \tag{10}$$

The weights $\bar{Q}$ and $\bar{R}$ (which are different from the weights Q and R in the objective function (1)) essentially parameterize the feedback gain matrices $\bar{F}$ that we consider. The performance of a candidate controller $\bar{F}$ in terms of the objective (1) is evaluated by conducting a closed-loop experiment and computing the cost from experimental data $\{y_k\}$, $\{u_k\}$. In an iterative procedure, the weights $\bar{Q}$ and $\bar{R}$ that achieve minimal cost are sought.

In order to facilitate a dimensionality reduction of the design problem, we consider parameterizations of the design weights $\bar{Q} = \bar{Q}(\theta)$ and $\bar{R} = \bar{R}(\theta)$ by a parameter vector $\theta \in \Theta \subseteq \mathbb{R}^p$. The space Θ of feasible parameter values is such that the LQR design (10) is well defined (i.e. $\bar{Q}(\theta) \geq 0$ and $\bar{R}(\theta) > 0$). With these parameterizations, the design task can then be formalized as the optimization problem

$$\min_{\theta \in \Theta} \hat{J}\big(\mathrm{lqr}(\bar{A}, \bar{B}, \bar{Q}(\theta), \bar{R}(\theta))\big), \tag{11}$$

where $\hat{J}$ is the finite length approximation (4) of (1).

Since the process (3) is affected by process noise, $\hat{J}$ is a random variable and (11) is a stochastic optimization problem. In principle, any method for stochastic optimization can be used to solve (11). Herein, we use the *Simultaneous Perturbation Stochastic Approximation* (SPSA) algorithm (presented in the next subsection) as a relatively straightforward and computationally lightweight way to address (11). The iterative numerical optimization of (11) constitutes the self-tuning LQR algorithm: in each optimization step i, the parameter vector θ^i is updated, which corresponds to an updated controller gain $\bar{F}^i$ through (10).

Remark 1. We reemphasize that the weights Q and R in (1) remain unchanged. The weights $\bar{Q}$ and $\bar{R}$ solely serve to compute LQR gains, while (1) with Q and R is used to evaluate the performance of the obtained controllers. To emphasize this difference, we refer to Q and R as the *performance weights* and to $\bar{Q}$ and $\bar{R}$ as the *design weights.*

Remark 2. In general, one cannot expect to recover the optimal gain (5) by solving (11) since the minimization in (11) is constrained, firstly, by parameterizing $\bar{F}$ with $\bar{Q}$ and $\bar{R}$ and, secondly, by parameterizing $\bar{Q}$ and $\bar{R}$ with θ. Parameterizing the controller gain by $\bar{Q}$ and $\bar{R}$ is useful because it guarantees that the obtained gain stabilizes the nominal system (6). The parameterization of $\bar{Q}(\theta)$ and $\bar{R}(\theta)$ by θ is helpful to reduce the dimensionality of the optimization problem. Section 4 presents an example where it is indeed possible to recover the optimal gain despite these parameterizations.

3.3 Stochastic Optimization Algorithm

In this section, we address the optimization problem (11), which is rewritten using $\hat{J}(\theta) = \hat{J}(\mathrm{lqr}(\bar{A}, \bar{B}, \bar{Q}(\theta), \bar{R}(\theta)))$ (with slight abuse of notation) as

$$\min_{\theta \in \Theta} \hat{J}(\theta). \tag{12}$$

The optimization problem at hand has the following characteristics: the objective function $\hat{J}(\theta)$ is stochastic; function evaluations are costly (one evaluation requires an experiment on the physical system); and no gradient information is available. Simultaneous Perturbation Stochastic Approximation (SPSA) is a stochastic optimization technique that is suitable for this class of problems (see Spall [2003]).

SPSA is based on forming approximations of the objective function gradient $\partial \hat{J}/\partial \theta$ from evaluations of $\hat{J}$ and updating the parameter in negative direction of the approximate gradient. We apply the SPSA algorithm as in [Spall, 2003, Cha. 7] to (12). Below, we give a brief summary of the algorithm; for further details and the theoretical underpinnings, the reader is referred to the original reference.

The basic mechanism that underlies the SPSA is the update equation for the parameter θ:

$$\theta^{i+1} = \theta^i - a^i\, g^i(\theta^i), \tag{13}$$

where θ^i is the parameter vector at iteration $i \geq 0$, $a^i > 0$ determines the step size, and $g^i(\theta^i)$ is an approximation of the gradient $\partial \hat{J}/\partial \theta$ at θ^i (superscript notation is used throughout for i). The gradient is approximated by computing the difference quotient at two points that are perturbations of θ^i along a random direction:

$$\begin{bmatrix} \frac{\hat{J}(\theta^i + c^i \Delta^{i,j}) - \hat{J}(\theta^i - c^i \Delta^{i,j})}{2c^i \Delta_1^{i,j}} \\ \vdots \\ \frac{\hat{J}(\theta^i + c^i \Delta^{i,j}) - \hat{J}(\theta^i - c^i \Delta^{i,j})}{2c^i \Delta_p^{i,j}} \end{bmatrix} =: g^{i,j}(\theta^i), \tag{14}$$

where $c^i > 0$ controls the step size of the gradient approximation, and $\Delta^{i,j}$ is a random perturbation vector generated from a symmetric ±1 Bernoulli distribution (each element $\Delta_\ell^{i,j}$ is either 1 or -1 with probability 0.5). Thus, the gradient approximation (14) requires two cost evaluations at $\theta^i \pm c^i \Delta^{i,j}$. In order to improve the gradient approximation, we take the average over n_g such gradient computations; that is, $g^i(\theta^i) = \sum_{j=1}^{n_\mathrm{g}} g^{i,j}(\theta^i)/n_\mathrm{g}$. Hence, one SPSA iteration (13) requires $2n_\mathrm{g}$ cost evaluations.

The step size sequences a^i and c^i decay according to $a^i = a/(i+1+\bar{a})^\alpha$ and $c^i = c/(i+1)^\gamma$, where $\alpha, a, \gamma, c > 0$, and $\bar{a} \geq 0$ are tuning parameters. Practical design guidelines for their selection (which typically need to be adjusted for the specific problem) are provided in [Spall, 2003, Sec. 7.5.2].

The constraint $\theta \in \Theta$ in the minimization (12) is dealt with by projecting θ^{i+1} in (13) to the feasible region Θ if it lies outside. Similarly, $\theta^i \pm c^i \Delta^{i,j}$ in (14) are projected to the feasible region if necessary before evaluating $\hat{J}$.

We typically initialize the SPSA with θ^0 such that $\bar{Q}(\theta^0) = Q$ and $\bar{R}(\theta^0) = R$; that is, we start with the initial design

(9). The optimization is terminated after a fixed number of iterations $i_{\max}$, or when the cost $\hat{J}$ does not improve over a number of iterations.

Remark 3. Notice that the SPSA algorithm does not require the cost $\hat{J}$ to be evaluated at a current iterate θ^i. According to (14), $\hat{J}$ is only evaluated at random locations around θ^i. The $2n_\mathrm{g}$ cost evaluations at every iteration provide a rough indication for $\hat{J}(\theta^i)$.

3.4 Noisy State Measurements

In this subsection, we briefly discuss how the self-tuning LQR approach can be used for the case of noisy state measurements; that is, where instead of (7),

$$y_k = x_k + v_k, \tag{15}$$

with $v_k \in \mathbb{R}^n$ i.i.d. noise with $\mathbb{E}[v_k] = 0$ and $\mathrm{Var}[v_k] = V_\mathrm{n}$.

For this case, the LQR is no longer the optimal controller (which is given by the combination of the LQR with a Kalman filter as state estimator, cf. Anderson and Moore [2007]). Despite not being optimal, a static feedback policy (2) may still be preferable over a dynamic controller for ease of implementation and low computational complexity. The self-tuning LQR approach can again be used to find improved controller gains. In fact, the approach is helpful even if the true system model is known, since the gain (5) does not represent the best static gain for the case of noisy state measurements (see Sec. 4.5 for further discussion).

The only modification to the self-tuning LQR approach required for noisy state measurements (15) instead of (7) is that the empirical cost (4) is evaluated with $\{y_k\}$ instead of $\{x_k\}$. That is, $\hat{J}$ is redefined as

$$\hat{J} = \frac{1}{K}\sum\nolimits_{k=0}^{K-1} y_k^\mathrm{T} Q y_k + u_k^\mathrm{T} R u_k. \tag{16}$$

Notice that it does not make a difference whether (1) or $\lim_{K\to\infty} \frac{1}{K}\mathbb{E}\,[\sum_{k=0}^{K-1} y_k^\mathrm{T} Q y_k + u_k^\mathrm{T} R u_k]$ is minimized since $\mathbb{E}\,[y_k^\mathrm{T} Q y_k + u_k^\mathrm{T} R u_k] = \mathbb{E}\,[(x_k+v_k)^\mathrm{T} Q (x_k+v_k) + u_k^\mathrm{T} R u_k] = \mathbb{E}\,[x_k^\mathrm{T} Q x_k + u_k^\mathrm{T} R u_k] + \mathbb{E}\,[v_k^\mathrm{T} Q v_k]$ and $\mathbb{E}\,[v_k^\mathrm{T} Q v_k] = \mathrm{tr}(Q V_\mathrm{n})$ is constant.

3.5 The Complete Algorithm

The complete self-tuning LQR algorithm is summarized in Alg. 1; it is applicable for both cases of perfect state measurements (7) and noisy state measurements (15). Algorithm 1 terminates after a fixed number of iterations $i_{\max}$; of course, other termination conditions can be used instead (for example, related to the change of $\hat{J}$). For clarity, we omitted the projection step in lines 7, 9, and 14, which ensures θ^+, θ^-, $\theta^{i+1} \in \Theta$.

4. ILLUSTRATIVE EXAMPLE

We illustrate the self-tuning LQR approach in this section through simulations, where the experiment in line 20 of Alg. 1 is replaced by simulations of the system (3). The simulation results highlight, in particular, the potential of the self-tuning LQR approach to yield the optimal controller F^* despite only having access to a perturbed system model $(\bar{A}, \bar{B})$. Furthermore, we illustrate the statistical properties of the underlying stochastic optimization by carrying out multiple tuning experiments.

Algorithm 1 Self-tuning LQR.

```
 1: initialize $\theta^0$ such that $\bar{Q}(\theta^0) = Q$, $\bar{R}(\theta^0) = R$
 2: for $i = 0$ to $i_{\max}-1$ do                      ▷ $i_{\max}$ SPSA iterations
 3:     $a^i \leftarrow a/(i+1+\bar{a})^\alpha$
 4:     $c^i \leftarrow c/(i+1)^\gamma$
 5:     for $j = 1$ to $n_\mathrm{g}$ do                 ▷ $n_\mathrm{g}$ gradient computations
 6:         draw $\Delta$ from symmetric $\pm 1$ Bernoulli distribution
 7:         $\theta^+ \leftarrow \theta^i + c^i\Delta$
 8:         $\hat{J}(\theta^+) \leftarrow$ CostEvaluation($\theta^+$)
 9:         $\theta^- \leftarrow \theta^i - c^i\Delta$
10:         $\hat{J}(\theta^-) \leftarrow$ CostEvaluation($\theta^-$)
11:         $g^{i,j}(\theta^i) \leftarrow$ Equation (14)          ▷ gradient computation
12:     end for
13:     $g^i(\theta^i) \leftarrow$ Average($g^{i,1}(\theta^i), \ldots, g^{i,n_\mathrm{g}}(\theta^i)$)
14:     $\theta^{i+1} \leftarrow \theta^i - a^i\, g^i(\theta^i)$
15: end for
16: LQR design: $\bar{F}^{\mathrm{final}} \leftarrow \mathrm{lqr}(\bar{A}, \bar{B}, \bar{Q}(\theta^{i_{\max}}), \bar{R}(\theta^{i_{\max}}))$
17: return $\bar{F}^{\mathrm{final}}$

Function CostEvaluation($\theta$)
18: LQR design: $\bar{F} \leftarrow \mathrm{lqr}(\bar{A}, \bar{B}, \bar{Q}(\theta), \bar{R}(\theta))$
19: update state feedback law (2) with $F = \bar{F}$
20: perform experiment and record $\{y_k\}$, $\{u_k\}$
21: return $(\sum_{k=0}^{K-1} y_k^\mathrm{T} Q y_k + u_k^\mathrm{T} R u_k)/K$
```

For the simulation study, we use a linearized model of a torque-actuated single-link inverted pendulum, which can be regarded as an abstraction of the experimental platform presented in Sec. 5 (problem parameters such as physical constants, noise properties, etc. are chosen to represent the true situation). MATLAB files to run the self-tuning algorithm for the simulation example of this section are provided as supplementary files with this paper. [1]

4.1 A Single-Link Inverted Pendulum

We consider a simple inverted pendulum given by a point-mass on a mass-less rod that is linked to the ground through a frictionless joint with one actuated rotational degree of freedom (DOF). The linearized dynamics about the vertical, unstable equilibrium are given by

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix} u(t), \tag{17}$$

where $x(t) = (\varphi(t), \dot{\varphi}(t))$ is the system state ($\varphi(t)$ pendulum angle in rad), $u(t)$ is the torque applied to the pendulum (Nm), $m = 5.3\,\mathrm{kg}$ is the pendulum mass, $\ell = 0.65\,\mathrm{m}$ is its length, and $g = 9.81\,\mathrm{m/s^2}$ is the gravity constant.

We discretize (17) assuming a zero-order hold and sampling time of $T_\mathrm{s} = 1/100\,\mathrm{s}$ to get the true system (3) with

$$A = \begin{bmatrix} 1.00 & 0.01 \\ 0.15 & 1.00 \end{bmatrix}, \quad B = \begin{bmatrix} 0.02 \\ 4.47 \end{bmatrix} \cdot 10^{-3}. \tag{18}$$

For the process noise variance, we choose $W_\mathrm{n} = \sigma_\mathrm{n}^2 B B^\mathrm{T}$ with $\sigma_\mathrm{n}^2 = 0.001$. We first treat the case of perfect state measurements (7), and the case of noisy measurements (15) is briefly discussed in Sec. 4.5. For this simulation study, simulations of (3), (18) yield the system trajectories $\{y_k\}$, $\{u_k\}$, which would be obtained from an experiment in reality (cf. Alg. 1, l. 20). All simulations are performed over a time horizon of 300 s, which is significantly larger than the system's time constants and representative for the time horizon used in the experiments in Sec. 5.

[1] Download at `http://www.cube.ethz.ch/downloads` or contact the first author.

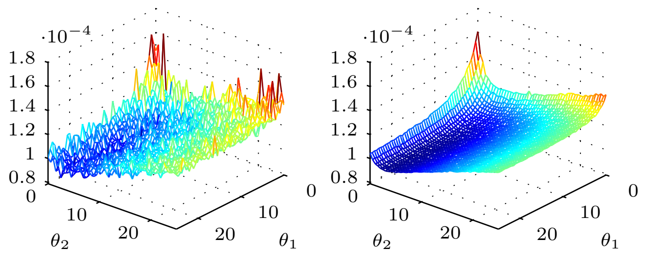

Fig. 1. Visualization of the cost function $\hat{J}(\theta)$ for the simulation example. LEFT: cost at each grid point evaluated once; RIGHT: cost averaged over 100 evaluations.

To reflect the situation in reality, we assume that we do not have the true model (3), but only a nominal model (6) at our disposal for designing LQR controllers. We represent our imperfect knowledge of the true model by reevaluating (17) with mass and length parameters underestimated by 20%. The numerical values for the nominal model (6) are

$$\bar{A} = \begin{bmatrix} 1.00 & 0.01 \\ 0.19 & 1.00 \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} 0.04 \\ 8.72 \end{bmatrix} \cdot 10^{-3}. \tag{19}$$

We seek to design a state-feedback controller that minimizes the cost (1) with $Q = 100\,I$ and $R = 1$.

4.2 Self-Tuning LQR Design

Next, we choose the design weights $\bar{Q}(\theta)$ and $\bar{R}(\theta)$, and the SPSA parameters. For the design weights, we consider a diagonal structure and choose

$$\bar{Q}(\theta) = \mathrm{diag}(100\theta_1, 100\theta_2), \quad \bar{R}(\theta) = 1 \tag{20}$$

with $\theta_1 \geq 0$ and $\theta_2 \geq 0$ to ensure $\bar{Q}(\theta) \geq 0$. Negative parameter values occurring at ll. 7, 9, 14 of Alg. 1, which would violate a constraint, are set to a small positive number. The parameters are initialized as $\theta^0 = (1, 1)$ such that $\bar{Q}(\theta^0) = Q$ and $\bar{R}(\theta^0) = R$.

The SPSA parameters were chosen according to the design guidelines referenced in Sec. 3.3; except for the parameter c, which was adjusted to obtain robust gradient approximations. We chose $n_{\mathrm{g}} = 2$ and $i_{\max} = 20$, which amount to 80 cost evaluations for the self-tuning experiment.

4.3 Analysis of the Underlying Optimization Problem

Ideally, we would like the self-tuning LQR approach to result in the LQR gain and cost that we would obtain if we knew the true model (18): $F^* = \mathrm{lqr}(A, B, Q, R) = (-67.48, -19.67)$ and $J^* = J(F^*) = 9.454 \cdot 10^{-5}$.

Figure 1 visualizes the cost function $\hat{J}(\theta)$ of the optimization (11) with parameterization (20). The graphs were obtained by uniformly griding the parameter space, computing the controller (10) with (20) for each grid point, and evaluating the cost (4) from simulation data.

The empirical cost of the initial design (9) is $\hat{J}_{1000}(\theta^0) = 1.23{\cdot}10^{-4}$ (average over 1000 evaluations), which is clearly suboptimal compared to J^*. The minimum is attained at $\theta^* = (20.61, 2.845)$ with a cost of $\hat{J}_{1000}(\theta^*) = 9.455{\cdot}10^{-5}$ and corresponding LQR gain $\bar{F}^* = \mathrm{lqr}(\bar{A}, \bar{B}, \bar{Q}(\theta^*), \bar{R}(\theta^*)) = (-67.48, -19.67)$. That is, the minimum cost and associated gain match those found with access to the true model. Hence, in this example, there is the potential to *perfectly* recover the optimal LQR controller despite imperfect model knowledge by choosing $\bar{Q}(\theta^*)$ and $\bar{R}(\theta^*)$ in the design (10). Due to the stochastic nature of the optimization problem (cf. Fig. 1), however, we cannot expect the optimizer to arrive at the optimal cost J^* in a finite number of iterations. Nonetheless, the stochastic optimizer typically yields a significant improvement over the initial design as the next subsection shows.

4.4 Self-Tuning Results

We executed the self-tuning LQR algorithm 1000 times on the problem of Sec. 4.1 with the algorithm settings described in 4.2. In particular, the algorithm was terminated after a fixed number of $i_{\max} = 20$ iterations. In order to evaluate the performance of each resulting controller $\bar{F}^{\mathrm{final}} = \bar{F}^{20}$, we simulated each one 100 times in closed-loop on the true system (3) and averaged the obtained costs $\hat{J}$. A histogram of the resulting empirical costs $\hat{J}_{100}(\theta^{20})$ for the 1000 controllers is given in Fig. 2.

The results show that the self-tuning algorithm resulted in improved controllers for each run. The median of the 1000 runs is $9.888 \cdot 10^{-5}$; that is, in 50% of the experiments we were within 5% of the optimal cost.

Clearly, we expect the self-tuning algorithm to yield better results if we are willing to spend more effort in terms of cost evaluations. For example, for $i_{\max} = 50$ and $n_{\mathrm{g}} = 10$ (1000 cost evaluations), and all other parameters unchanged, the median of the final cost $\hat{J}_{100}(\theta^{50})$ is $9.697 \cdot 10^{-5}$.

4.5 Noisy State Measurements

If one has noisy state measurements (i.e. (15) instead of (7)), static state feedback does no longer represent the optimal controller structure, but it may still be preferable for practical considerations. In this situation, the self-tuning approach can be used to find an improved gain F over the LQR design (5). This is true even for the hypothetical case of a perfect model ($\bar{A} = A$ and $\bar{B} = B$), since the LQR design is suboptimal due to imperfect state measurements. In practice, one typically has a combination of both imperfect model knowledge and imperfect state measurements such as in the experiment discussed in the next section.

5. EXPERIMENTS ON AN INVERTED PENDULUM

We used the inverted pendulum in Fig. 3 as a testbed to demonstrate the performance of the self-tuning LQR algorithm in an experimental setting.

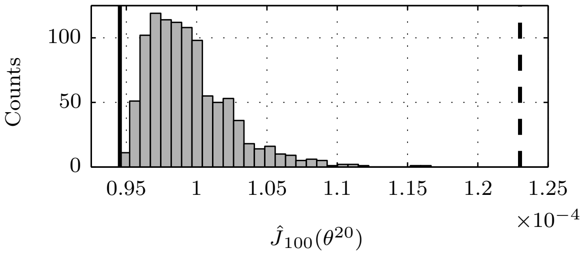

Fig. 2. The outcome of 1000 executions of the self-tuning LQR algorithm on the illustrative example. The thick dashed and solid black lines represent the cost $J(\theta_0)$ of the initial design (9) and the optimal cost J^*.

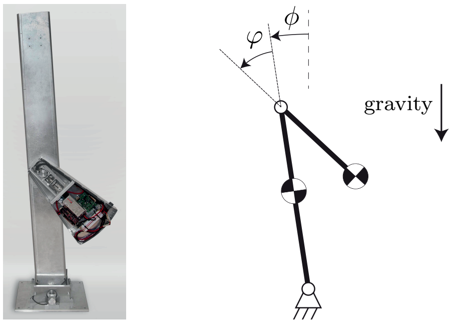

Fig. 3. The inverted pendulum (left) and an abstraction as a point-mass system (right) depicting the system's two rotational degrees of freedom ϕ and φ.

5.1 System Description and Linear Model

The inverted pendulum (Fig. 3 (left)) can be abstracted as two rigid bodies linked by revolute joints: the pendulum body, which is linked to the ground with one rotational DOF, is kept in an upright position through appropriate motion of the actuated second body. The second body (called *module*) carries sensors, a computer, a motor, and a battery. Trimpe and D'Andrea [2012] use six of these modules to balance a cubic structure on any one of its corners or edges. The system studied here can be regarded as a one dimensional abstraction of this balancing cube.

The two DOFs of the system are parametrized by ϕ (rad) and φ (rad) as depicted in Fig. 3 (right). The motor unit implements a velocity feedback controller that tracks module angular velocity commands u_k. An LQR controller is to be designed (and automatically tuned) that computes motor commands u_k that stabilize the pendulum about the equilibrium configuration of an upright pendulum ($\phi = 0$) and downward module ($\varphi = 0$).

The linearized dynamics of the system about the equilibrium are given by

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}u_k, \tag{21}$$

$$\tilde{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -2\cdot 10^{-4} & -0.001 & 1.001 & 0.01 \\ -0.03791 & -0.1171 & 0.1297 & 1.001 \end{bmatrix}, \; \tilde{B} = \begin{bmatrix} 0.01 \\ 1 \\ 0.00117 \\ 0.1169 \end{bmatrix},$$

with state $\tilde{x}_k = (\varphi_k, \dot{\varphi}_k, \phi_k, \dot{\phi}_k)$ and sampling time 0.01 s. The model captures the dynamics of the pendulum including the velocity feedback on the motors; for details refer to Trimpe and D'Andrea [2009].

To compensate for steady-state offsets, we augment the system with an integrator state z_k:

$$x_{k+1} = \begin{bmatrix} \tilde{x}_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} \tilde{A} & 0 \\ [T_s \;\; 0 \;\; 0 \;\; 0] & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_k \\ z_k \end{bmatrix} + \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} u_k. \tag{22}$$

The model (22) is used as the nominal model (6) for computing the LQR gains (10). The augmented integrator state is implemented as a controller state.

All relevant states can be measured on the experimental platform: φ_k and ϕ_k are measured by angle encoders, and measurements of $\dot{\phi}_k$ are obtained from rate gyro sensors on the pendulum body. Measuring the module velocity $\dot{\varphi}_k$ is not required since, by the high-gain assumption of the velocity control loop, we can use the previous command u_{k-1} as an approximation for $\dot{\varphi}_k$ (see Trimpe and D'Andrea [2009] for details). Since the sensor measurements are corrupted by sensor noise (especially the rate gyro), we have the case of noisy state measurements (15).

We expect the self-tuning LQR approach to yield improvements, since 1) we have imperfect state measurements, and 2) the model (21) is an approximation of the true dynamics.

5.2 Self-Tuning LQR Design

For the LQR design, we use a modification of the cost (4) to include weights on state-input cross terms:

$$\hat{J} = \frac{1}{K}\sum_{k=0}^{K-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + 2x_k^{\mathrm{T}} N u_k \tag{23}$$

with $Q = \operatorname{diag}(1,1,1,1,0.01)$, $R = 1.5$, and $N = (0,-1,0,0,0)$. Trimpe and D'Andrea [2009, 2012] show that this non-zero weight on state-input cross terms corresponds to penalizing the difference of two successive input commands ($u_k - u_{k-1}$). They found the resulting controllers to yield satisfactory balancing performance in practice, which is why we keep the same structure of the cost matrices herein. For the self-tuning LQR algorithm, this modification of the cost function makes no essential difference; (23) is simply used in place of (4) when evaluating the experimental performance. Also, the LQR gain for the generalized cost can readily be computed using standard LQR design tools (such as `dlqr` in MATLAB). For the generalized LQR problem, one has the constraint $Q - NR^{-1}N^{\mathrm{T}} \geq 0$ in addition to $R > 0$. The cost (23) is computed from experimental data. Possible DC components on state and input trajectories (especially the integrator state) are removed prior to evaluating (23).

We choose a three dimensional parameterization for the design weights used in the self-tuning mechanism: $\bar{Q}(\theta) = \operatorname{diag}(\theta_1, \theta_2, \theta_1, \theta_2, 0.01)$, $\bar{R}(\theta) = \theta_3$, and $\bar{N}(\theta) = N$. With this parameterization, we allow the optimizer to trade off the deviation in position states (through variation of θ_1), the deviation in velocity states (θ_2), and the control effort (θ_3) with each other. The integrator state, which is nonphysical and its contribution to the overall cost small, was deemed unimportant and its corresponding weight left unparameterized.

The SPSA parameters used in Alg. 1 were chosen according to the practical design guidelines discussed in Sec. 3.3, with the exception of a and c. Parameters a and c were adjusted, following a number of controller evaluations around the optimization starting point θ^0, to give both robust gradient approximations and a reasonable initial step length for the optimization parameter. Whenever necessary, the projection of parameters $\theta \notin \Theta$ to the feasible region Θ (i.e. such that $\bar{Q}(\theta) - \bar{N}(\theta)\bar{R}(\theta)^{-1}\bar{N}(\theta)^{\mathrm{T}} \geq 0$ and $\bar{R}(\theta) > 0$) was done by a subordinate optimization finding the closest point in Θ in Euclidean distance.

5.3 Experimental Results

This section details the results of an experimental run of the self-tuning approach on the inverted pendulum system. The high level portions of the self-tuning LQR algorithm (controller design and stochastic optimization)

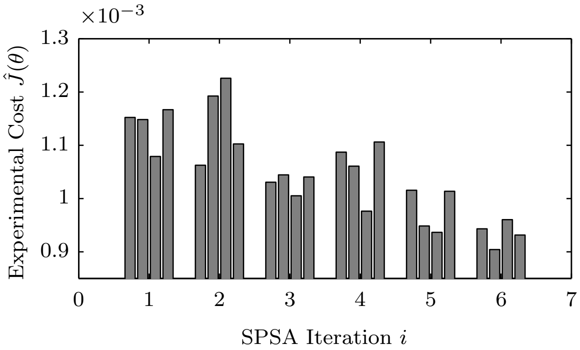

Fig. 4. Evolution of the self-tuning experiment on the inverted pendulum.

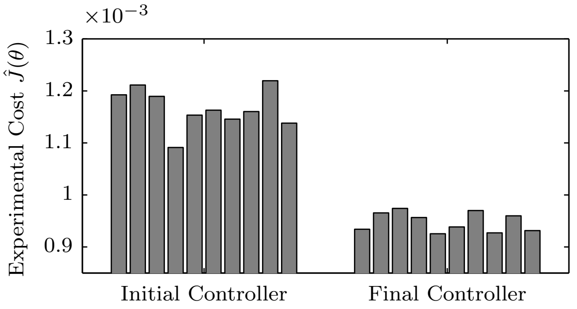

Fig. 5. Tests of the controller resulting from the self-tuning experiment on the inverted pendulum. The bar diagram shows the costs of ten evaluations of the final controller $\bar{F}^{\text{final}}$, in comparison to ten experiments with the initial design $\bar{F}^0$.

were run on a desktop workstation, while the real-time code implementing the controller (8) ran on a single-board computer on the module. The experiment was terminated after a fixed number of $i_{\max} = 6$ iterations (24 cost evaluations/balancing experiments) and, θ was initialized such that $\bar{Q}(\theta^0) = Q$, $\bar{R}(\theta^0) = R$ and $\bar{N}(\theta^0) = N$.

Fig. 4 shows the cost evaluations over the experimental run of the self-tuning algorithm. Note that per iteration i, four cost function evaluations occur (two per gradient approximation). Accordingly, the figure shows four bars centered around each iteration number. Each of these evaluations is generated by a parameter value which is perturbed from the current iterate θ^i (see Remark 3). The average of the four bars can be regarded as an approximation of the cost at θ^i, and the graph shows a general downward trend in controller cost. Superimposed on that trend are the effects of stochastic noise in the controller evaluations.

In order to quantify the outcome of the self-tuning experiment, the resulting final controller $\bar{F}^{\text{final}}$ was evaluated ten times in balancing experiments which were separate to the learning process. The resulting costs $\hat{J}$ are shown in Fig. 5, in comparison to ten evaluations of the initial controller $\bar{F}^0$. The average of the ten evaluations is 9.48e^{-4} for the final controller, and 11.7e^{-4} for the initial controller, which corresponds to a cost reduction of 19%.

6. CONCLUDING REMARKS

The self-tuning LQR approach presented in this paper mimics a process often conducted by the designer of an LQR controller: the weighting matrices of the quadratic cost function underlying the design are varied iteratively such that the controller's performance in experiments on the physical plant improves. The presented approach is a general tuning method that is applicable (in principle) in any practical situation where one seeks to automatically tune a static feedback controller. As an example where tuning is useful, we consider the situation where the model of a linear process to be controlled is known only approximately. For this case, we show through simulations that the self-tuning approach can potentially recover the optimal LQR cost and gain (i.e. the ones that would be obtained with perfect model knowledge), and thus a separate system identification step can be avoided. The effectiveness of the self-tuning LQR approach was also demonstrated in an experimental setting on an inverted pendulum.

A drawback of the current tuning method is the critical dependence of its performance on the parameters for the SPSA optimizer, which is an inherent feature of this optimization technique (Spall [2003], for instance, gives only rough guidelines for the choice for parameters, which typically have to be adapted for a specific problem). Furthermore, the optimizer only takes the latest cost evaluations into account for making a controller update (instead of the entire history of cost evaluations). For these reasons, function approximation techniques that successively form an estimate of the entire cost function $\hat{J}(\theta)$ may be an attractive alternative to stochastic optimization and thus an interesting subject for future investigations.

REFERENCES

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

K. J. Åström and B. Wittenmark. *Adaptive Control.* Dover, 2nd edition, 2008.

D. W. Clarke, P. P. Kanjilal, and C. Mohtadi. A generalized LQG approach to self-tuning control – Part I. Aspects of design. *International Journal of Control*, 41 (6):1509–1523, 1985.

M. J. Grimble. Implicit and explicit LQG self-tuning controllers. *Automatica*, 20(5):661–669, 1984.

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*, pages 310–317, April 2011.

J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control.* Wiley, 2003.

S. Trimpe and R. D'Andrea. A limiting property of the matrix exponential with application to multi-loop control. In *Proc. of the 48th IEEE Conf. on Decision and Control and 28th Chinese Control Conf.*, pages 6419–6425, Shanghai, P.R. China, December 2009.

S. Trimpe and R. D'Andrea. The Balancing Cube: A dynamic sculpture as test bed for distributed estimation and control. *IEEE Control Systems Magazine*, 32(6):48–75, December 2012.