Tuning LQR controllers: a sensitivity-based approach

Daniele Masti, Mario Zanon, and Alberto Bemporad

***Abstract*— We introduce an approach to efficiently tune LQR controllers for linear time-invariant systems to match a prescribed closed-loop behavior, such as the one given by a reference model. The proposed approach is able to efficiently tune the LQR controller, even for high dimensional systems and is superior in terms of achieved tracking performance and other criteria with respect to global optimization methods commonly used for black-box, simulation-based, automated tuning.**

***Index Terms*— Identification for control; machine learning.**

I. Introduction

THE linear quadratic regulator (LQR) is a well-known and established technique, which has been successfully applied in a very large number of industrial applications. Yet, despite its natural support for multi-input/multi-output (MIMO) systems, and the the numerous extensions to augment its capabilities (such as model predictive control [1] (MPC) and LQG/LTR [2]), its potential spread in applications is hampered by the lack of well-assessed tuning procedures required to reach satisfactory performance [3]–[5].

The factors that make tuning LQR controllers a possibly challenging task are many. The first one is that the number of "tuning-knobs" scales quadratically with the number of involved states/inputs. Another factor is that *loop-shaping* approaches often used to tune linear controller in the frequency domain are not directly applicable to LQR, which is based on a performance index defined in the time-domain.

For the above reasons, the development of auto-tuning procedures for LQR, i.e., techniques that are able to automatically select the weight matrices defining the infinite-horizon performance index, has been the subject of numerous works based on different definitions of "performance". For instance, in [6] a rule-based approach to tune predictive controllers for servomechanisms is presented. Parametric approaches based on differentiating the optimality conditions of optimization-based finite-horizon controllers have also been the object of numerous works (see for, instance [7]–[9]) and have shown promising results. In [10] two multi-objective approaches based on lexicographic and hierarchic optimization algorithms are also presented for MPC controllers. In [11], [12], two solutions are discussed to match an existing controller to a set of weight matrices. In the context of infinite-horizon LQR, in [4], [13], [14] rule-based approaches are presented to tune LQR controllers for systems with a specific number of states and inputs. In [15] a method to synthesize an LQR-based PID regulator for DC-DC converters using iterative-learning is presented.

Derivative-free, black-box, global optimization approaches have also been proposed for automatic tuning of controllers. In [5], an approach for tuning predictive controllers also in relation to the available computational power is discussed, while in [16] a method based on Gaussian processes is presented for tuning LQR controllers. We also mention [17], which proposes a preference-based approach that requires *no* quantification of a closed-loop performance measure.

All the aforementioned methods present practical limitations. Although they require very few assumptions, global optimization methods often suffer from the curse of dimensionality [18], [19], which limits their applicability to auto-tune only a small number of parameters, and therefore to small-dimensional systems. When applicable, *derivative-based methods* enjoy better scaling properties, yet most works are strictly meant for *finite-horizon* optimal control formulations as they rely on inferring information from the optimality conditions associated with the computed control action to proceed in the tuning process. On the other hand, the intrinsic closed-loop stability properties of the *infinite-horizon* formulation are often desirable to have. Rule-based strategies are instead often limited in their scope.

In this letter, we detail a derivative-based approach to systematically tune LQR controllers. The main idea is to design *offline* the controller so that the closed-loop system will behave as much as possible to a user-provided (possibly nonlinear) reference model and, in the particular context of this work, its reference-to-output (I/O) behavior. This approach is attractive because it can be well adapted to time-domain formulations; it does not require a baseline controller to imitate, and also does not require an explicit parameterization of the reference model as it can rely solely on desired closed-loop trajectories. In developing our technique, we also surpass the limitations of competing approaches —which often consider only a subset of possible LQR parameterizations— in a way that does not require a nonlinear programming solver able to handle semi-definite constraints (NLSDP), while maintaining desirable scalability properties that derivative-free methods lack.

This work is related to the *inverse reinforcement learning* [20], *apprenticeship learning* [21] and *learning from demonstration* [22] methods, in which reward functions to

The authors are with IMT School for Advanced Studies, Piazza San Francesco 19, Lucca, Italy. email: {daniele.masti, mario.zanon, alberto.bemporad}@imtlucca.it.

This paper was partially supported by the Italian Ministry of University and Research under the PRIN'17 project "Data-driven learning of constrained control systems", contract no. 2017J89ARP.

imitate a known agent are sought from recorded state/input trajectories. Compared to them, we do not require that the "expert demonstration" is performed on the actual target system and solve the problem in remarkably different fashion. The problem we aim to solve is also similar to the one faced in [23], where genetic algorithms are used to tune a continuous-time discounted LQR controller, and bi-level programming approaches (such as [24]) in which optimal control schemes are tuned to reproduce human behavior.

The paper is organized as follows: in Section II we formally state the problem, introduce the relevant notation, the approach we rely on to efficiently compute derivatives, and some approaches to parameterize the weights matrices that do not require one to impose semidefiniteness constraints. In Section III we provide a set of experiments in which we show the capabilities of the proposed approach. We finally draw some conclusions in Section IV.

II. Problem statement

We consider LQR controllers based on a Linear Time Invariant (LTI) discrete-time[1] dynamical system of the form

$$\Sigma \triangleq \begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{cases} \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, $y_k \in \mathbb{R}^{n_y}$ denote the states, control inputs, and outputs respectively. A, B, C are *known* matrices of appropriate dimensions (for example, obtained from an identification procedure or from linearization of a nonlinear model at a certain operating point). We further assume that Σ is controllable, and its state x_k to be measurable to avoid introducing state observers in our formulation.

Let us assume that we have collected a dataset (r_k, w_k) of desired closed-loop behaviors, $k = 0, \dots, \mathcal{H}$, where r_k is the reference signal and w_k is the corresponding desired output of the system we wish to get under closed-loop control. A special case of the above assumption is to assume that w_k is generated by a reference model

$$\Sigma_{\text{RM}} \triangleq \begin{cases} \zeta_{k+1} &= f_{RM}(\zeta_k, r_k) \\ w_k &= h_{RM}(\zeta_k) \end{cases} \tag{2}$$

where $\zeta_k \in \mathbb{R}^{n_z}$, $r_k \in \mathbb{R}^{n_y}$, $w_k \in \mathbb{R}^{n_y}$ and f_{RM} and h_{RM} are possibly nonlinear maps. For convenience, in this paper we will focus on such a model-reference framework to address the controller tuning problem, that is to tune an LQR controller such that $y_k \approx w_k$ when excited by the same reference signal r_k. Consider the following *offline* problem

$$\min_{K,F,x,y} \quad \mathcal{L}(y,w) \tag{3a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + B(-Kx_k + Fr_k), \qquad k \in \mathbb{I}_0^{\mathcal{H}} \tag{3b}$$

$$y_k = Cx_k, \qquad k \in \mathbb{I}_0^{\mathcal{H}} \tag{3c}$$

where $c \triangleq (c_0, \dots, c_{\mathcal{H}})$, $c \in \{x, y, w, r, u\}$, and $\mathbb{I}_a^b$ is the set of all integers in the interval $[a, b]$. Note that r and w are given data of the problem. We want to stress thatΣ_{RM} is not necessarily needed, as long as its sampled input-output behavior is available. This allows, e.g., to try reproducing complex behaviors for which no model is available, such as human experts manually controlling the process (human drivers, pilots, process-control operators, etc.).

[1]The extension to the continuous-time case follows using the same arguments.

We assume that the sequence r, w covers the entire range of behaviors of interest. Clearly, that the quality of the results of the method proposed in this paper will depend on how rich and how long the training signals are. Correctly designing the experiment to fit (r_k, w_k), however, is non trivial [25] and beyond the scope of this work.

A simple choice to attempt making $y_k \approx w_k$ in closed loop is to minimize the simple quadratic cost function

$$\mathcal{L}(y,w) = \sum_{k=0}^{\mathcal{H}} \|y_k - w_k\|_2^2, \tag{4}$$

although other types of functions might be used too.

Remark 1: The problem stated in (3) optimizes both K and F. Had Σ_{RM} been known, this could have been avoided as the optimal F would have been a function of Σ and K itself.

A. LQR feedback parameterization

Directly solving (3) entails solving an optimization problem over the space of all possible linear feedback gains, including non-stabilizing ones. Instability is clearly undesirable from a design point of view and can be problematic from a numerical point of view. We propose to tackle that issue by restricting the domain of K to the set of stabilizing feedback laws. In particular, we will focus on K which are the solution of an LQR problem. Note that this does not entail any loss of generality, as any stabilizing feedback law can be obtained from a suitably defined LQR formulation [12].

Choosing this parameterization has several practical advantages. In particular, the desirable stability properties that full state LQR controllers may deliver for some classes of tuning parameters [26], [27]. Additionally, in case the introduction of constraints is of interest, the obtained LQR weight matrices can be used to formulate an MPC controller which maintains the same closed-loop behavior of the unconstrained controller for small signals [11], [12], i.e., when constraints are inactive.

In general, the LQR problem is formulated as

$$\begin{aligned} \min_{u} \quad & \sum_{k=0}^{\infty} \begin{bmatrix} x_k \\ u_k \end{bmatrix}' H \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ \text{s.t.} \quad & \text{Equation (1)}, \\ & x_0 = \mathbf{x_0}, \end{aligned} \tag{5}$$

where $H \triangleq \begin{bmatrix} Q & N \\ N' & R \end{bmatrix} \succ 0$, and $Q = Q' \in \mathbb{R}^{n_x \times n_x}$, $R = R' \in \mathbb{R}^{n_u \times n_u}$, $N \in \mathbb{R}^{n_x \times n_u}$ and $\mathbf{x_0}$ is the initial state of the system. The general solution for this problem is given by a static state feedback gain, namely $u_k = -Kx_k$, where the gain $K \in \mathbb{R}^{n_u \times n_x}$ is obtained by computing the only stabilizing solution to the Discrete Algebraic Riccati Equation (DARE)

$$S = A'SA + Q - (A'SB + N)K, \tag{6a}$$

$$K = (B'SB + R)^{-1}(B'SA + N'). \tag{6b}$$

As S is an implicit functions of H, in the following we summarize the computation of K through (6) as $K = f_K(H)$.

B. Model reference via LQR matching

The problem of synthesizing an LQR controller to match a given closed-loop behavior (r_k, w_k) can be recast into the following optimization problem

$$\begin{aligned}
\min_{H,K,F,x,y} \quad & \mathcal{L}(y,w) && && \text{(7a)}\\
\text{s.t.} \quad & x_{k+1} = Ax_k + B(-Kx_k + Fr_k), && k \in \mathbb{I}_0^{\mathcal{H}}, && \text{(7b)}\\
& y_k = Cx_k, && k \in \mathbb{I}_0^{\mathcal{H}}, && \text{(7c)}\\
& K = f_K(H), && && \text{(7d)}\\
& H = H' \succ 0. && && \text{(7e)}
\end{aligned}$$

Note that, by exploiting the results of [12], one could equivalently formulate the problem by removing variable H, constraint (7d), and by imposing the stability conditions that $A - BK$ has all its eigenvalues strictly inside the unit circle and relying on the methods proposed in [12] to recover H.

Problem (7) is a NLSDP problem, since constraints (7b) and (7d) define nonlinear equality constraints and (7e) is a semidefinite constraint. Since all functions are differentiable, (7) can be solved by derivative-based optimization algorithms. The main non-standard features of the problem are constraint (7d) and (7e).

C. Derivatives of implicit functions

The main difficulty with (7d) is that f_K is an implicit function. In order to tackle this issue, let us rewrite (6) as

$$S = A'SA + Q - (A'SB + N)(B'SB + R)^{-1}(B'SA + N'). \tag{8}$$

While several numerical methods are available to solve the DARE (6), they do not directly provide the required derivatives. In order to address this issue, we exploit the implicit function theorem [28], [29]. For ease of notation, we define the vectorized matrices S, H as s, h, respectively, and (8) as $f(s,h) = 0$, such that $\frac{\mathrm{d}s}{\mathrm{d}h} = -\left(\frac{\partial f(s,h)}{\partial s}\right)^{-1} \frac{\partial f(s,h)}{\partial h}$.

The derivative of f_K with respect to H is then directly obtained by applying the chain rule. We ought to stress that we intentionally eliminate S and use a standard DARE solver in order to guarantee that the unique stabilizing solution is obtained at each iterate performed by the NLP solver. However, an alternative approach could consist in introducing S as an optimization variable, with (6) directly as a constraint replacing (7d) in Problem (7). However this would be dangerous because one does not have the a-priori guarantee that the solver does not converge to one of the (in general infinitely many) destabilizing solutions of the DARE.

D. Enforcement of the semidefinite constraint

Despite its simple nature, the presence of (7e) requires one to employ a NLSDP solver to seek the solution of (7). This is however problematic because there are very few well tested and maintained NLSDP solvers available in literature, and implementing such a solver from scratch is far from trivial. In this work we take a more practical approach and reformulate (7) so that a standard nonlinear programming (NLP) solver can be instead used[2]. In particular, we present two alternative formulations which differ by how matrix H is parameterized:

1. **Direct Cholesky factorization [30], [31].** Because H is a positive definite matrix, it can be uniquely factorized as $H = L'L$,where L is an upper triangular matrix with positive diagonal elements. This means that one can replace H (as decision variables) with the vector θ of the nonzero elements of its Cholesky factor L.
2. **An *indirect* approach based on LDL' factorization.** A positive definite matrix H can be factorized [32] as $H = \bar{L}D\bar{L}'$, where $\bar{L} \in \mathbb{R}^{n_x+n_u \times n_x+n_u}$ is a lower triangular matrix in which the diagonal elements are equal to 1, and $D \in \mathbb{R}^{n_x+n_u \times n_x+n_u}$ is a diagonal matrix with only positives entries. Accordingly, a way to enforce (7e) is to impose that the entries of the factor D of H stay positive [33], namely:
$$D_{ii} > 0, \; i = 1, \ldots, n_x + n_u, \tag{9}$$
where D_{kj} stands for the entry of D in the $j-$th column an $k-$th row. In this case, since H is symmetric, the decision variables vector θ will only have to contain the elements of its lower (upper) half.

The first approach replace (7e) with a nonlinear parameterization with bound constraints on some elements of θ. This is advantageous because many optimization algorithms support only this kind of bounds. The indirect LDL' criterion retains a linear parameterization of the elements of H, but it requires introducing a set of nonlinear constraints.

Remark 2: The LDL' decomposition could have also been used to provide a *direct* parameterization of the matrix H. Such an approach would be very similar to the already proposed Cholesky parameterization and its analysis is therefore omitted.

Remark 3: To employ the proposed criteria in our scheme, we need to compute the sensitivities $\frac{\partial D_{ii}}{\partial H}$, which requires us to differentiate through the LDL' decomposition. Indeed, this is not a concern as this computation is a well-understood procedure which is also natively supported by many automatic differentiation packages.

Remark 4: The condition (9) is equivalent to state that a Cholesky factorization of H exists.

In the following we will analyze the performance of both the proposed reinterpretation of the indirect LDL' criterion and the Cholesky parameterization as substitute for the LMI constraint (7e).

E. Formulatation of the tuning problem as NLP

Let us LQR controller tuning problem as

$$\begin{aligned}
\min_{\theta,x,y,F,K,H} \quad & \mathcal{L}(y,w) & \\
\text{s.t.}: \quad & x_{k+1} = Ax_k + B(-Kx_k + Fr_k), & k \in \mathbb{I}_0^{\mathcal{H}},\\
& y_k = Cx_k, & k \in \mathbb{I}_0^{\mathcal{H}},\\
& K = f_{\mathrm{K}}(H), H = f_H(\theta), & \\
& g(\theta) \geq 0. &
\end{aligned} \tag{10}$$

[2]Analyzing which approach is the best is outside the scope of this paper.

Here, for the Cholesky parameterization we have that $g(\theta) \geq 0$ stands for $\theta_{\bar{k}+i} - \varepsilon \geq 0,\ i \in \mathbb{I}_1^{n_x+n_u}$, $\bar{k} = \frac{(n_x+n_u-1)(n_x+n_u)}{2}$, and $f_H(\theta) = L(\theta)'L(\theta)$, with

$$L(\theta) = \begin{bmatrix} \theta_{\bar{k}+1} & \theta_0 & \cdots & \theta_{n_u+n_x-1} \\ 0 & \theta_{\bar{k}+2} & \theta_{n_u+n_x} & \cdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \theta_{\bar{k}+n_u+n_x} \end{bmatrix}. \quad (11)$$

For the indirect LDL' parameterization, $g(\theta) \geq 0$ stands for the constraints described in (9), and

$$f_H(\theta) = X(\theta)' + X(\theta), \quad (12)$$

where $X(\theta)$ is, without loss of generality, an upper triangular matrix. In both cases, $\varepsilon \ll 1$ is a constant used to recast the strict positivity constraints as non-strict ones, as usual in optimization.

Remark 5: The optimization problems (10) admit infinite equivalent solutions [12]. For this reason, it might be useful to introduce regularizers to improve numerical conditioning.

Remark 6: Additional constraints on H can also be imposed to promote specific properties of the solution (e.g.: sparsity).

III. Numerical experiments

We now report the results obtained by the presented approaches on various tests. The implementation was carried out using MATLAB's `fmincon` interior-point (IP) solver is used to solve the NLP and CasADi [34] is used as automatic-differentiation package. For comparison with global optimization approaches, we also report the results obtained using MATLAB's particle swarm solver (PSO) [35].

We design r as a sequence of step signals with random amplitude drawn from a uniform distribution $\mathcal{U}(-1.5, 1.5)$ with a period of 60 steps, collecting 300 samples in total. IP results were obtained setting as starting condition a $H(\theta)$ equal to the identity matrix. For the PSO, to smooth the optimization procedure, we set a limit of 280 iterations a maximum absolute value of 10 on the elements of θ, and further impose a minimum value of 10^{-4} on the diagonal elements of H in the LDL' case. Since MATLAB `particleswarm` solver does not support nonlinear constraints, we handle them as penalties in the objective function.

In all cases, the cost function (4) is used to assess quality of closed-loop performance plus a small regularization term $\lambda\|H(\theta)\|_F^2$ involving the Frobenius norm of H, with $\lambda = 2 \cdot 10^{-5}$. In this particular set of examples, without loss of generality, we further imposed $x_0 = 0$. All experiments are performed on a PC with an Intel Core i7 4770k with 16GB of RAM.

A. Experiments with reference models impossible to achieve

We consider two dynamical systems: an unstable system with non-minimum-phase zeros, and a stable minimum-phase one. In both cases, the aim is to obtain a closed-loop behavior as similar as possible to the one of an LTI reference model that is however impossible to exactly achieve due to the performance limitation imposed by the delays (in the latter case) and by the non-minimum phase components (in the former case).

1) Minimum phase system: We consider as system Σ the minimal representation of the following transfer function

$$G(z) = \frac{z^2 - z + 0.16}{z^4 - 1.09z^3 + 0.176z^2 + 0.1051z - 0.02604}, \quad (13)$$

where z^{-1} is the unit-delay operator. In this example, we consider as reference model Σ_{RM} the system

$$\Sigma_{RM}(z) \triangleq \begin{cases} \zeta_{k+1} &= 0.15\zeta_k + 0.85r_k \\ w_k &= \zeta_k \end{cases} \quad (14)$$

The results obtained for the above settings are reported in Table I, where we can observer that the two parameterizations achieve similar results, yet a much greater computational cost is necessary for the Cholesky parameterization (11). Nevertheless, our approach is able to reach better results while requiring fewer function evaluations than the PSO-based approach. The achieved closed-loop behavior is reported in Figure 1 for the Cholesky case. There we can appreciate that, except for the first time-step where the different behavior is due to the intrinsic 2-step delay of the original system Σ, there is good correspondence between the behavior of the controlled system Σ_{LQR} and the one of reference model Σ_{RM}. Results for the indirect LDL' case are indistinguishable and are therefore not reported.

2) Unstable non-minimum phase system: As test system Σ we consider a minimal realization of the following transfer function

$$H(z) = \frac{z^2 + 0.9z - 0.9}{z^3 - 0.98z^2 + 0.7076z + 0.7238}, \quad (15)$$

which has two unstable poles in $z_{1,2} = 0.74 \pm j\sqrt{0.9}$ and an unstable zero in $z = -1.5$. Also for this system we consider the reference model given in (14). The comparison between

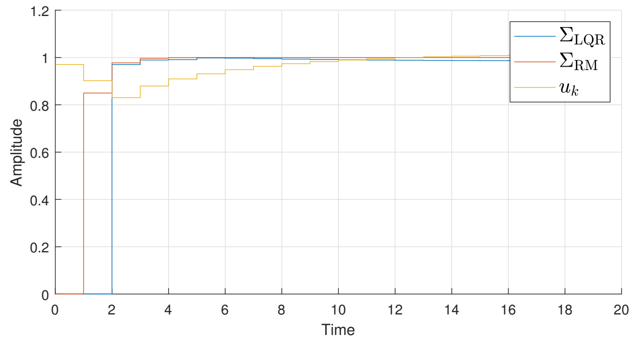

Fig. 1: Closed loop behavior of the controlled system Σ_{LQR} in comparison to Σ_{RM} and $H(z)$ (13). Results obtained using the Cholesky parameterization (11).

Method	Iterations	# function evaluations	Best cost	CPU time (seconds)
IP+LDL' (12)	220	405	5.66	6
IP+Cholesky (11)	597	731	5.66	10
PSO+LDL' (12)	88	8900	6.03	65
PSO+Cholesky (11)	280	28100	5.70	296

TABLE I: Achieved cost and computational effort of the two proposed parameterizations for the system in (13).

the best-achieved cost for the two parameterizations is reported in Table II. There we can see a similar situation to the one found in the previous case: the IP solver was able to achieve a very good fit using both parameterizations and, for the indirect LDL' case, fewer iterations were required to reach convergence. The PSO solver was also able to achieve very good results, yet it could not surpass our proposed approach despite the enormous amount of function evaluations required. The achieved closed-loop behavior is shown in Figure 2 for the LDL' parameterization. Results for the Cholesky case are indistinguishable and omitted.

B. Trading off tracking and control effort

The proposed approach can be extended to tune LQR controllers w.r.t. other targets in addition to the sole model-reference tracking. Consider for instance tuning a controller to trade off between tracking and energy/amplitude of the control action. This can be simply obtained by adding a penalty on the input signal to the cost function $\mathcal{L}(\cdot,\cdot)$. To demonstrate this, consider the following minimum-phase dynamical system Σ

$$J(z) = \frac{0.1429z^2 - 0.1429z + 0.02286}{z^3 - 1.4z^2 + 0.61z - 0.084}, \tag{16}$$

and the reference model given in (14). For this system, it is indeed possible to achieve a perfect tracking of the reference model, as shown in Figure 3. In there, however, we can also observe that such a perfect tracking of Σ_{RM} requires a great feedback action effort $u_k^{\mathrm{F}} = -K_A x_k$. Assume now that we would like to limit control action $|u_k^{\mathrm{F}}| \leq 1$ when tracking a unit-step reference signal from zero starting condition. To do so, we employ the loss function $\bar{\mathcal{L}}(y,w,u) = \mathcal{L}(y,w) + \sum_{k=0}^{\mathcal{H}} \eta(\max\{1,|u_k^{\mathrm{F}}|\}^2 - 1)$, where $\eta \geq 0$, and repeat the tuning process using a square wave of amplitude 1 and 60 steps period as test signal. The results obtained with $\eta = 0.02$ are also shown in Figure 3. There we observe how the new closed loop $\Sigma_{LQR/B}$ is still able to achieve an effective tracking, yet the $K_B x_k$ remains smaller than 1. The results in Figure 3 have been obtained using the Cholesky parameterization.

Method	Iterations	# function evaluations	Best cost	CPU time (seconds)
IP+LDL' (12)	206	320	0.942	4
IP+Cholesky (11)	583	759	0.942	9
PSO+LDL' (12)	280	28100	0.962	241
PSO+Cholesky (11)	280	28100	0.947	297

TABLE II: Achieved cost and computational effort of the two proposed parameterizations for the system in (15).

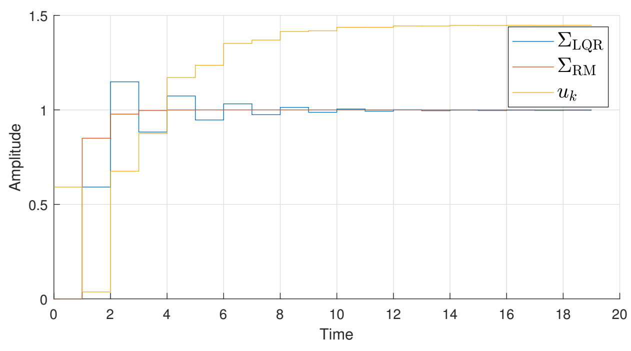

Fig. 2: Closed-loop behavior of the controlled system Σ_{LQR} in comparison to Σ_{RM} for (15). Results were obtained using the indirect LDL' parameterization (12).

C. Hammerstein-Wiener system

We investigate the behavior of the tuning approach when trying to reproduce a nonlinear system. To this end we use the plant described in (16) and takes a reference model the following Hammerstein-Wiener system

$$\Sigma_{RM}^{HW}(z) \triangleq \begin{cases} \zeta_{k+1} = 0.15\zeta_k + 0.85(r_k + \frac{r_k^2 \mathrm{sign}(r_k)}{4}) \\ w_k = \zeta_k + \tanh(\frac{\zeta_k}{2}) \end{cases} \tag{17}$$

The results obtained using the Cholesky parameterization are shown in Figure 4. Indistinguishable performance were obtained also using the indirect LDL' approach. There we can appreciate that even for this nonlinear benchmark the proposed approach is able to obtain good set-point tracking performance.

In Figure 4 we also observe the step response of an output error model of order $[4,4,1]$ identified by Matlab's System Identification Toolbox [36] on the same (r_k, w_k) dataset. The good resemblance between Σ_{LQR} and the identified model suggests that our tuning approach achieves a result very close to the best tracking performance that a linear system can possibly achieve w.r.t Σ_{RM}^{HW}.

D. Computational performance for medium-sized systems

To better assess the scalability of the proposed approach we analyze the CPU time required to tune a controller for a random system with 30 stable poles and 29 minimum phase zeros. To tune the involved parameters, the solver required 520 seconds and 4094 function evaluations for the Cholesky case, and 308 seconds and 3145 evaluations for the LDL' case. The optimal cost $\mathcal{L}^\star$ in both cases is less than 10^{-3}, i.e., an almost perfect tracking. Interestingly, the PSO fails in both cases to synthesize a controller with comparable performance even after 280 iterations.

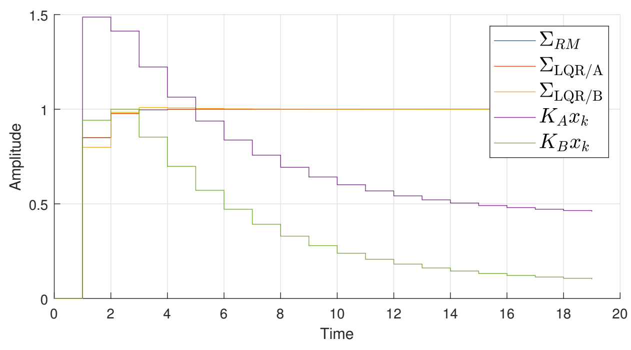

Fig. 3: Comparison between $\Sigma_{\mathrm{LQR/A}}$ and $\Sigma_{LQR/B}$ and the reference Σ_{RM}. The first two systems represent the closed-loop obtained setting, respectively, $\eta = 0$ and $\eta = 0.02$ in the tuning procedure. $K_A x_k$ and $K_B x_k$ represent the controller feedback action for $\Sigma_{\mathrm{LQR/A}}$ and $\Sigma_{LQR/B}$ respectively.

IV. Conclusions

We introduced a numerically efficient method to automatically tune LQR controllers to match a user-provided I/O behavior. Our approach reaches better results using a fraction of the CPU time required by PSO, also for medium-scale systems with 30 poles. We have also shown how the proposed approach can be easily modified to take into account additional goals such as avoiding input saturation.

Future works will explore extensions to uncertain and stochastic systems, in addition to study convergence properties and sample complexity [37], also in comparison with competing approaches.

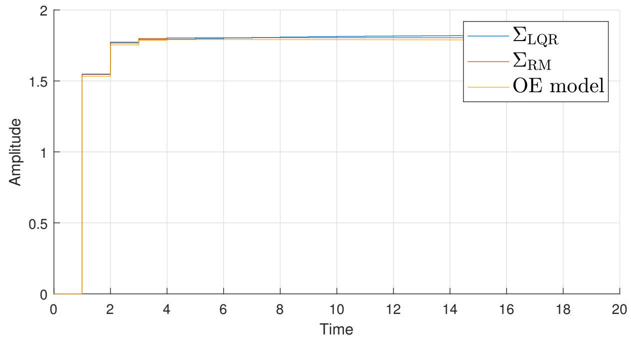

Fig. 4: Closed-loop behavior of the controlled system $\Sigma_{\rm LQR}$ in comparison to $\Sigma_{\rm RM}$, for the Hammerstein-Wiener system.

References

[1] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[2] M. Athans, "A tutorial on the LQG/LTR method," in *1986 American Control Conference*, pp. 1289–1296, IEEE, 1986.

[3] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[4] V. Kumar E and J. Jerome, "Algebraic Riccati equation based Q and R matrices selection algorithm for optimal LQR applied to tracking control of 3rd order magnetic levitation system," *Archives of Electrical Engineering*, vol. 65, no. 1, pp. 151–168, 2016.

[5] M. Forgione, D. Piga, and A. Bemporad, "Efficient Calibration of Embedded MPC," in *Proc. of the 21st IFAC World Congress*, 2020.

[6] E. J. Davison, D. E. Davison, and R. Milman, "Transient response shaping, model based cheap control, saturation indices and MPC," *European journal of control*, vol. 11, no. 4-5, pp. 288–300, 2005.

[7] A. Agrawal, S. Barratt, S. Boyd, and B. Stellato, "Learning convex optimization control policies," in *Learning for Dynamics and Control*, pp. 361–373, PMLR, 2020.

[8] S. Gros and M. Zanon, "Data-Driven Economic NMPC Using Reinforcement Learning," *IEEE Transactions on Automatic Control*, vol. 65, pp. 636–648, Feb 2020.

[9] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Transactions on Automatic Control*, pp. 1–1, 2020.

[10] A. Yamashita, A. Zanin, and D. Odloak, "Tuning of model predictive control with multi-objective optimization," *Brazilian Journal of Chemical Engineering*, vol. 33, no. 2, pp. 333–346, 2016.

[11] S. Di Cairano and A. Bemporad, "Model predictive control tuning by controller matching," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 185–190, 2009.

[12] M. Zanon and A. Bemporad, "Constrained control and observer design by inverse optimality," *arXiv preprint arXiv:2003.10166*, 2020.

[13] S. Doddabasappa, "LQR control design for a DC-DC converter using sensitivity functions," Master's thesis, The Pennsylvania State University, 2019.

[14] J.-B. He, Q.-G. Wang, and T.-H. Lee, "PI/PID controller tuning via LQR approach," *Chemical Engineering Science*, vol. 55, no. 13, pp. 2429–2439, 2000.

[15] O. Saleem and M. Rizwan, "Performance optimization of LQR-based PID controller for DC-DC buck converter via iterative-learning-tuning of state-weighting matrix," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 32, no. 3, p. e2572, 2019.

[16] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," in *2016 IEEE Int. Conf. on robotics and automation (ICRA)*, pp. 270–277, IEEE, 2016.

[17] M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," in *Proc. of the 2021 European Control Conference (ECC)*, 2021. To appear. arXiv preprint arXiv:2003.11294.

[18] R. Moriconi, M. P. Deisenroth, and K. Kumar, "High-dimensional bayesian optimization using low-dimensional feature spaces," *arXiv preprint arXiv:1902.10675*, 2019.

[19] S. Chen, J. Montgomery, and A. Bolufé-Röhler, "Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution," *Applied Intelligence*, vol. 42, no. 3, pp. 514–526, 2015.

[20] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. of 17th Int. Conf. on Machine Learning*, 2000.

[21] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of the 21st Int. Conf. on Machine Learning*, p. 1, 2004.

[22] M. Palan, S. Barratt, A. McCauley, D. Sadigh, V. Sindhwani, and S. Boyd, "Fitting a linear control policy to demonstrations with a Kalman constraint," in *Learning for Dynamics and Control*, pp. 374–383, PMLR, 2020.

[23] H. El-Hussieny and J.-H. Ryu, "Inverse discounted-based LQR algorithm for learning human movement behaviors," *Applied Intelligence*, vol. 49, no. 4, pp. 1489–1501, 2019.

[24] S. Albrecht, *Modeling and numerical solution of inverse optimal control problems for the analysis of human motions*. PhD thesis, Technische Universität München, 2013.

[25] A. Tsiamis and G. J. Pappas, "Linear systems can be hard to learn," *arXiv preprint arXiv:2104.01120*, 2021.

[26] C. Chen, *On the robustness of the linear quadratic regulator via perturbation analysis of the Riccati equation*. PhD thesis, Dublin City University, 2015.

[27] U. Shaked, "Guaranteed stability margins for the discrete-time linear quadratic optimal regulator," *IEEE Transactions on Automatic Control*, vol. 31, no. 2, pp. 162–165, 1986.

[28] M. C. Priess, R. Conway, J. Choi, J. M. Popovich, and C. Radcliffe, "Solutions to the inverse LQR problem with application to biological systems analysis," *IEEE Transactions on control systems technology*, vol. 23, no. 2, pp. 770–777, 2014.

[29] T.-C. Kao and G. Hennequin, "Automatic differentiation of Sylvester, Lyapunov, and algebraic Riccati equations," *arXiv preprint arXiv:2011.11430*, 2020.

[30] J. C. Pinheiro and D. M. Bates, "Unconstrained parametrizations for variance-covariance matrices," *Statistics and computing*, vol. 6, no. 3, pp. 289–296, 1996.

[31] S. Burer and R. D. Monteiro, "A projected gradient algorithm for solving the maxcut SDP relaxation," *Optimization methods and Software*, vol. 15, no. 3-4, pp. 175–200, 2001.

[32] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. JHU press, 2013.

[33] R. J. Vanderbei and H. Y. Benson, "On formulating semidefinite programming problems as smooth convex nonlinear optimization problems," tech. rep., Princeton University, 2000.

[34] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[35] The MathWorks, Inc., *Optimization Toolbox*. United States, 2021.

[36] The MathWorks, Inc., *System Identification Toolbox*. United States, 2021.

[37] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *Int. Conf. on Machine Learning*, pp. 1467–1476, PMLR, 2018.