# Efficient model-free Q-factor approximation in value space via log-sum-exp neural networks

# Efficient model-free Q-factor approximation in value space via log-sum-exp neural networks

Giuseppe C. Calafiore, Corrado Possieri

*Abstract*— **We propose an efficient technique for performing data-driven optimal control of discrete-time systems. In particular, we show that log-sum-exp (LSE) neural networks, which are smooth and convex universal approximators of convex functions, can be efficiently used to approximate Q-factors arising from finite-horizon optimal control problems with continuous state space. The key advantage of these networks over classical approximation techniques is that they are convex and hence readily amenable to efficient optimization.**

## I. Introduction

Optimal control is concerned with determining the *best* control strategy that minimizes a certain optimization criterion [1], [2]. This branch of control theory finds several applications spanning from medicine [3] to energy systems [4].

When the dynamics of the plant to be controlled are known, the classical approach to solve a finite-horizon optimal control problem is the so called *Dynamic Programming Algorithm*. Although such an algorithm provides an exact solution to the optimal control problem, it may be extremely time consuming, even for medium-sized problems, [5]. For this reason a continuing research effort has been put in determining approximate solutions to optimal control problem. For instance, [6] proposes an approach based on optimal control and decision theory, [7] introduces a policy iteration method based on temporal differences, [8] provides a reinforced learning approach; see also [9, Sec. 1.5] and references therein. Another remarkable technique is the *Approximate Dynamic Programming*, which, by using simplified models and approximations, allows one to tackle large scale, stochastic decision processes [10], [11].

The approach we are proposing in this paper consists in approximating the Q-factors of the dynamic programming problem via a novel type of convex neural networks called LSE-networks, introduced in [12]. The method of approximating the Q-factors via linear and nonlinear functions has been proven to be considerably successful; see, e.g., [13] and references therein. Furthermore, approximating unknown, possibly non-convex, functions via convex approximators has been proved successful to solve several classes of optimization problems; see, e.g., [14] for the approximation capabilities of convex algebraic polynomials.

The key advantage of approximating the Q-factors via LSE networks rather than with classical techniques is that the former are capable of approximating any convex function while possessing the additional and desirable property of being themselves smooth and convex functions, a feature which makes them optimized efficiently. In particular, these networks allow one to perform the optimization on-line, without requiring any approximation in policy space.

*Structure of the paper:* In Section II, the dynamic programming algorithm is briefly reviewed following the exposition in [9]. In Section III, it is shown how the Q-factors arising from such an algorithm can be approximated via LSE-networks. Numerical experiments showing the effectiveness of such a procedure are reported in Section IV. Finally, conclusions are drawn in Section V.

## II. The dynamic programming algorithm

Consider the discrete-time system

$$x_{k+1} = f_k(x_k, u_k), \tag{1}$$

where $k \in \mathbb{N}$ is a time index, $x_k \in \mathbb{R}^n$ is the state of the system at time $k$, $u_k \in \mathbb{R}^m$ is the control or decision variable, which has to be selected at each $k \in \mathbb{N}$ from the set $U_k(x_k)$ that depends on the current state $x_k$, and $f_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a state update map. To this system we associate a cost function of the form

$$J(x_0, u_0, \dots, u_{N-1}) = \sum_{k=0}^{N-1} g_k(x_k, u_k) + g_N(x_N), \tag{2}$$

which evaluates the performance of the sequence $\{u_0, \dots, u_{N-1}\}$ of control inputs when system (1) is initialized at $x_0$. Our objective is to determine an optimal sequence $\{u_0^\star, \dots, u_{N-1}^\star\}$ that minimizes the cost function (2) over all sequences $\{u_0, \dots, u_{N-1}\}$ satisfying the control constraint, thus obtaining

$$J^\star(x_0) := \min_{\substack{u_k \in U_k(x_k)\\ k=0,\dots,N-1}} J(x_0, u_0, \dots, u_{N-1}).$$

The approach that is classically employed for solving such a problem is the *Dynamic Programming Algorithm*, see, e.g., [15]. Such an algorithm constructs sequentially the *cost-to-go* functions $J_N^\star(x_N), J_{N-1}^\star(x_{N-1}), \dots, J_0^\star(x_0)$ starting from

$$J_N^\star(x_N) = g_N(x_N), \quad \forall x_N,$$

and, going backwards for $\kappa = N-1, \dots, 0$, letting, $\forall x_\kappa$,

$$J_\kappa^\star(x_\kappa) := \min_{u_\kappa \in U_\kappa(x_\kappa)} \left\{ g_\kappa(x_\kappa, u_\kappa) + J_{\kappa+1}^\star(f_\kappa(x_\kappa, u_\kappa)) \right\}. \tag{3}$$

G. C. Calafiore is with Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, 10129 Torino, Italy, and also with IEIIT-CNR Torino, 10129 Torino, Italy (e-mail: giuseppe.calafiore@polito.it).

C. Possieri is with Istituto di Analisi dei Sistemi ed Informatica “A. Ruberti”, Consiglio Nazionale delle Ricerche (IASI-CNR), 00185 Roma, Italy (e-mail: corrado.possieri@iasi.cnr.it).

By Bellman's principle of optimality [16], defining, for $\kappa = 0,\dots,N-1$, the tail cost function $J_\kappa(x_\kappa, u_\kappa,\dots,u_{N-1}) = \sum_{k=\kappa}^{N-1} g_k(x_k,u_k) + g_N(x_N)$, we have that

$$J^\star_\kappa(x_\kappa) =: \min_{\substack{u_k\in U_k(x_k)\\ k=\kappa,\dots,N-1}} J_\kappa(x_\kappa, u_\kappa,\dots,u_{N-1}). \tag{4}$$

Therefore, for every initial state $x_0$, the optimal cost $J^\star(x_0)$ equals $J^\star_0(x_0)$ that is obtained at the last step of the dynamic programming recursion. Once the functions $J^\star_N(x_N),\dots,J^\star_0(x_0)$ have been determined, the optimal control sequence can be computed as

$$u^\star_k \in \operatorname*{arg\,min}_{u_k\in U_k(x_k)} \left\{ g_k(x^\star_k, u_k) + J^\star_{k+1}(f_k(x^\star_k,u_k)) \right\},$$

where $x^\star_k$ is the solution to system (1) when the control sequence $\{u^\star_0,\dots,u^\star_{N-1}\}$ is applied. Note that the same algorithm can be used to solve the tail subproblem (4).

An equivalent formulation of the dynamic programming algorithm [17], [18] can be given in terms of the Q-factors

$$Q_\kappa(x_\kappa,u_\kappa) := g_\kappa(x_\kappa,u_\kappa) + J^\star_{\kappa+1}(f_\kappa(x_\kappa,u_\kappa)), \tag{5}$$

defined for $\kappa = 0,\dots,N$. Indeed, by (3), we have $J^\star_\kappa(x_\kappa) = \min_{u_\kappa\in U_\kappa(x_\kappa)} Q_\kappa(x_\kappa,u_\kappa)$, and hence the dynamic programming algorithm can be equivalently rewritten in terms of the Q-factors as

$$\begin{aligned} Q_\kappa(x_\kappa,u_\kappa) &= g_\kappa(x_\kappa,u_\kappa) \\ &\quad + \min_{u_{\kappa+1}\in U_{\kappa+1}(f(x_\kappa,u_\kappa))} Q_{\kappa+1}(f_\kappa(x_\kappa,u_\kappa),u_{\kappa+1}), \end{aligned}$$

for $\kappa = N-1,\dots,0$, starting with $Q_N(x_N,u_N) = g_N(x_N)$, $\forall x_N$. Once the Q-factors $Q_0(x_0,u_0),\dots,Q_{N-1}(x_{N-1},u_{N-1})$ have been determined, the one-step lookahead control $u^\star_k$ at time $k\in\{0,\dots,N-1\}$ can be obtained on-line as

$$u^\star_k \in \operatorname*{arg\,min}_{u_k\in U_k(x_k)} Q_k(x_k,u_k), \tag{6}$$

where $x_k$ denotes the state of system (1) at time $k$.

## III. Q-LEARNING VIA LSE NEURAL NETWORKS

In this section, we show how the Q-factors defined in (5) can be approximated via LSE neural networks. The key advantage of using these networks is that they synthesize a convex function and hence they are readily amenable to efficient optimization. In particular, if the function $Q_\kappa(x_\kappa,u_\kappa)$ is approximated by an LSE neural network (which synthesizes the function $\tilde{Q}_\kappa(x_\kappa,u_\kappa)$) and the set $U_k(x_k)$ is convex, then a solution to the minimization problem

$$\tilde{u}^\star_k \in \operatorname*{arg\,min}_{u_k\in U_k(x_k)} \tilde{Q}_k(x_k,u_k), \tag{7}$$

which has to be solved in order to determine an approximate $\tilde{u}^\star_k$ of $u^\star_k$, can be determined efficiently by using convex optimization tools [19], such as interior point algorithms [20] and disciplined convex optimization tools [21], [22]. On the other hand, minimizing a generic non-convex function usually involves compromises such as long computation time or sub-optimality of the solution [19]. Moreover, as classical Q-learning approaches [17], [18], the tool given here can be also used in a model-free setting; see also [23].

In Section III-A we briefly review LSE networks, whereas, in Section III-B, we show how these networks can be used to approximate the Q-factors. Finally, in Section III-C, we discuss the quality of the determined suboptimal control.

### A. LSE *neural network*

Let LSE (Log-Sum-Exp) be the class of functions $\ell : \mathbb{R}^n \to \mathbb{R}$ that can be written as

$$\ell(\xi) = \log\left(\sum_{k=1}^K b_k \exp(\alpha^{(k)\top}\xi)\right),$$

for some $K\in\mathbb{N}$, $b_k\in\mathbb{R}_{>0}$, $\alpha^{(k)} = [\,\alpha^{(k)}_1 \ \cdots \ \alpha^{(k)}_n\,]^\top \in \mathbb{R}^n$, $k=1,\dots,K$, where $\xi = [\,\xi_1 \ \cdots \ \xi_n\,]^\top$ is a vector of variables. Further, given $T\in\mathbb{R}_{>0}$, define the class $\mathrm{LSE}_T$ of functions $\ell_T : \mathbb{R}^n\to\mathbb{R}$ that can be written as

$$\ell_T(\xi) = T\log\left(\sum_{k=1}^K b_k^{1/T}\exp(\alpha^{(k)\top}\xi/T)\right),$$

for some $K\in\mathbb{N}$, $b_k\in\mathbb{R}_{>0}$, and $\alpha^{(k)}\in\mathbb{R}^n$, $k=1,\dots,K$. By letting $\beta_k \doteq \log b_k$, $k=1,\dots,K$, we have that functions in the family $\mathrm{LSE}_T$ can be equivalently parameterized as

$$\ell_T(\xi) = T\log\left(\sum_{k=1}^K \exp(\alpha^{(k)\top}\xi/T + \beta_k/T)\right),$$

where the $\beta_k$s have no sign restrictions. It may sometimes be convenient to highlight the full parameterization of $\ell_T$, in which case we shall write $\ell_T^{(\overrightarrow{\alpha},\beta)}$, where $\overrightarrow{\alpha} = (\alpha^{(1)},\dots,\alpha^{(K)})$, and $\beta = (\beta_1,\dots,\beta_K)$.

By [12], functions in $\mathrm{LSE}_T$ are smooth and convex. Furthermore, given a real-valued convex function $g(\cdot)$ defined on a compact convex set $\mathcal{K}\subset\mathbb{R}^n$, for all $\varepsilon>0$ there exists $T^\star>0$ such that, for all $T\in(0,T^\star)$, there exists a function $\ell_T\in\mathrm{LSE}_T$ such that

$$|\ell_T(\xi) - g(\xi)| \leqslant \varepsilon, \qquad \text{for all } \ \xi\in\mathcal{K},$$

i.e., functions in $\mathrm{LSE}_T$ are universal, convex, smooth approximators of convex functions over convex, compact sets; see Theorem 2 of [12]. Furthermore, as shown in [12], functions in $\mathrm{LSE}_T$ can be equivalently be represented as feedforward neural networks with one hidden layer having exponential activation in the hidden layer and logarithmic activation in the output neuron (see Fig. 1).

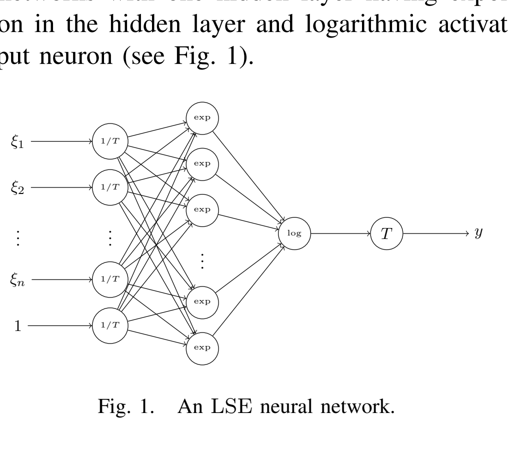

Fig. 1. An LSE neural network.

These networks can be trained both on-line and off-line by using classical algorithms, such as the Levenberg-Marquardt algorithm [24], the gradient descent with momentum [25], the Fletcher-Powell conjugate gradient [26], or the stochastic gradient descent [27]. Most of these algorithms make use of the gradients of the function synthesized by the network with respect to the parameters, which for LSE$_T$ network are

$$\nabla_{\alpha^{(i)}} \ell_T^{(\overrightarrow{\alpha},\beta)}(\xi) = \frac{\exp({\alpha^{(i)}}^\top \xi/T + \beta_i/T)\,\xi}{\sum_{k=1}^{K} \exp({\alpha^{(k)}}^\top \xi/T + \beta_k/T)},$$
$$\nabla_{\beta_i} \ell_T^{(\overrightarrow{\alpha},\beta)}(\xi) = \frac{\exp({\alpha^{(i)}}^\top \xi/T + \beta_i/T)}{\sum_{k=1}^{K} \exp({\alpha^{(k)}}^\top \xi/T + \beta_k/T)}.$$

### B. Convex approximation of the Q-factors

In this subsection, we show how the LSE networks reviewed in Section III-A can be used to approximate the Q-factors $Q_k(x_k, u_k)$. Assume to have at one's disposal *state-control-successor state* triples $(x_k^{(i)}, u_k^{(i)}, x_{k+1}^{(i)})$, where $i \in \{1, \dots, s\}$ denotes the experiment number, and $k \in \{0, \dots, N-1\}$ denotes the discrete-time in the $i$-th experiment. By considering the triplets $(x_{N-1}^{(i)}, u_{N-1}^{(i)}, x_N^{(i)})$, $i = 1, \dots, s$, define the corresponding sample Q-factor

$$\gamma_{N-1}^{(i)} := g_{N-1}(x_{N-1}^{(i)}, u_{N-1}^{(i)}) + g_N(x_N^{(i)}).$$

An LSE$_T$ approximate $\ell_T^{(\overrightarrow{\alpha}_{N-1},\beta_{N-1})}(x_{N-1}, u_{N-1})$ of the Q-factor $Q_{N-1}(x_{N-1}, u_{N-1})$ can be determined by designing the weights $\overrightarrow{\alpha}_{N-1}$ and $\beta_{N-1}$ so to minimize

$$\sum_{i=1}^{s} \delta_{N-1}^{(i)} \left\| \gamma_{N-1}^{(i)} - \ell_T^{(\overrightarrow{\alpha}_{N-1},\beta_{N-1})}(x_{N-1}^{(i)}, u_{N-1}^{(i)}) \right\| + R(\overrightarrow{\alpha}_{N-1}, \beta_{N-1}), \quad (8)$$

where $R(\cdot,\cdot)$ is a regularization term that is independent of $\gamma_N$, and $\delta_{N-1}^{(i)}$ is a weighting factor, $i = 1, \dots, s$. Note that the problem above can be addressed by using one of the algorithms recalled in Section III-A.

Once the weights $\alpha_N$ and $\beta_N$ have been determined, by considering the triplets $(x_{N-2}^{(i)}, u_{N-2}^{(i)}, x_{N-1}^{(i)})$, $i = 1, \dots, s$, define the corresponding approximate sample Q-factor

$$\gamma_{N-2}^{(i)} := g_{N-2}(x_{N-2}^{(i)}, u_{N-2}^{(i)}) + \min_{u_{N-1} \in U_{N-1}(x_{N-1}^{(i)})} \ell_T^{(\overrightarrow{\alpha}_{N-1},\beta_{N-1})}(x_{N-1}^{(i)}, u_{N-1}). \quad (9)$$

Note that, differently from $\gamma_{N-1}^{(i)}$ that are actually samples of the Q-factor $Q_{N-1}(x_{N-1}, u_{N-1})$, the values $\gamma_{N-2}^{(i)}$ are approximate samples of the Q-factor $Q_{N-2}(x_{N-2}, u_{N-2})$ due to the fact that $\ell_T^{(\overrightarrow{\alpha}_{N-1},\beta_{N-1})}$ is used rather than $Q_{N-1}(x_{N-1}, u_{N-1})$ to compute them. Furthermore, it is worth stressing that, differently from classical Q-learning approaches, the second term in the right-hand side (9) can be easily determined by using convex optimization tools.

Once the approximate samples $\gamma_{N-2}^{(i)}$ have been computed, an LSE$_T$ approximate $\ell_T^{(\overrightarrow{\alpha}_{N-2},\beta_{N-2})}(x_{N-2}, u_{N-2})$ of the Q-factor $Q_{N-2}(x_{N-2}, u_{N-2})$ can be determined by designing the weights $\overrightarrow{\alpha}_{N-2}$ and $\beta_{N-2}$ so to minimize

$$\sum_{i=1}^{s} \delta_{N-2}^{(i)} \left\| \gamma_{N-2}^{(i)} - \ell_T^{(\overrightarrow{\alpha}_{N-2},\beta_{N-2})}(x_{N-2}^{(i)}, u_{N-2}^{(i)}) \right\| + R(\overrightarrow{\alpha}_{N-2}, \beta_{N-2}),$$

where $\delta_{N-2}^{(i)}$ is a weighting factor, $i = 1, \dots, s$.

By iterating such a procedure backward, we obtain weights $\overrightarrow{\alpha}_\kappa$ and $\beta_\kappa$ such that the LSE function $\ell_T^{(\overrightarrow{\alpha}_\kappa,\beta_\kappa)}(x_\kappa, u_\kappa)$ is a convex, smooth approximate of the Q-factor $Q_\kappa(x_\kappa, u_\kappa)$, $\kappa = 0, \dots, N-1$. Hence, once these functions have been computed, an approximate optimal control $\tilde{u}_k^\star$ is given by (7) with $\tilde{Q}_\kappa(x_\kappa, u_\kappa) := \ell_T^{(\overrightarrow{\alpha}_\kappa,\beta_\kappa)}(x_\kappa, u_\kappa)$. It is worth pointing out that, if the set $U_\kappa(x_\kappa)$ is convex, the problem in (7) is convex and hence it can be solved efficiently on-line, without the need of an approximation in policy space.

Note that, as in classical data-driven approaches (see, e.g., the one given in [9, Sec. 2.1.4]), the technique given in this section to approximate Q-factors via convex function does not require a mathematical model of the system under consideration, but just samples of its trajectories.

It is worth noticing that, once a new set of triplets $(x_k^{(i+1)}, u_k^{(i+1)}, x_{k+1}^{(i+1)})$ is gathered (e.g., by using the approximate optimal controls obtained by solving (7)), $k = 0, \dots, N-1$, the approximate samples

$$\gamma_\kappa^{(i+1)} := g_\kappa(x_\kappa^{(i+1)}, u_\kappa^{(i+1)}) + \min_{u_{\kappa+1} \in U_{N-1}(x_{\kappa+1})} \ell_T^{(\overrightarrow{\alpha}_{\kappa+1},\beta_{\kappa+1})}(x_{\kappa+1}^{(i+1)}, u_{\kappa+1})$$

can be used to iteratively update the weights $\overrightarrow{\alpha}_\kappa$ and $\beta_\kappa$ by using, for instance, the stochastic gradient descent algorithm.

Finally, note that if the Q-factors $Q_\kappa(x_\kappa, u_\kappa)$ are convex, then by Proposition 4 of [12], they can be approximated with arbitrary precision by letting $K$ be sufficiently large and $T$ sufficiently small. On the other hand, if the Q-factors $Q_\kappa(x_\kappa, u_\kappa)$ are not convex, they can however be approximated in a neighborhood of their minimum by a convex function by suitably selecting the parameters $\delta_k^{(i)}$, thus making the proposed Q-learning approach suitable even in the non-convex case.

### C. Quality of the suboptimal control

A predictor of the quality of the suboptimal control that is usually employed in the literature [9] is the difference

$$Q_k(x_k, \tilde{u}_k^\star) - Q_k(x_k, u_k^\star),$$

where $u_k^\star$ is given by (6) and $\tilde{u}_k^\star$ is given by (7). If the Q-factor $Q_k(\cdot,\cdot)$ is a convex function, we can guarantee that this difference can be made arbitrarily small over a compact, convex set by letting $\tilde{Q}_k(x_k, u_k) = \ell_T^{(\overrightarrow{\alpha}_k,\beta_k)}(x_k, u_k)$. As a matter of fact, consider the quality index

$$\begin{aligned}&(Q_k(x_k, \tilde{u}_k^\star) - Q_k(x_k, u_k^\star)) + (\tilde{Q}_k(x_k, u_k^\star) - \tilde{Q}_k(x_k, \tilde{u}_k^\star)) \\ &= (\tilde{Q}_k(x_k, u_k^\star) - Q_k(x_k, u_k^\star)) - (\tilde{Q}_k(x_k, \tilde{u}_k^\star) - Q_k(x_k, \tilde{u}_k^\star))\end{aligned}$$

Since $u_k^\star$ and $\tilde{u}_k^\star$ are given by (6) and (7), we have that

$$Q_k(x_k, \tilde{u}_k^\star) - Q_k(x_k, u_k^\star) \geqslant 0,$$
$$\tilde{Q}_k(x_k, u_k^\star) - \tilde{Q}_k(x_k, \tilde{u}_k^\star) \geqslant 0,$$

for all admissible $x_k$. Hence, by the convex universal approximation theorem given in [12], it results that, given a convex compact set $\mathcal{C}$ such that $(x_k, u_k^\star, \tilde{u}_k^\star) \in \mathcal{C}$, there exists $T^\star \in \mathbb{R}_{>0}$ such that, for all $T \in (0, T^\star)$, there exists a function $\ell_T^{(\overrightarrow{\alpha}_k, \beta_k)}(x_k, u_k)$ in $\mathrm{LSE}_T$ such that, if $\tilde{Q}_k(x_k, u_k^\star) = \ell_T^{(\overrightarrow{\alpha}_k, \beta_k)}(x_k, u_k)$, then $(\tilde{Q}_k(x_k, u_k^\star) - Q_k(x_k, u_k^\star)) - (\tilde{Q}_k(x_k, \tilde{u}_k^\star) - Q_k(x_k, \tilde{u}_k^\star)) < \varepsilon$, for all $(x_k, u_k^\star, \tilde{u}_k^\star) \in \mathcal{C}$. Hence, we have that

$$Q_k(x_k, \tilde{u}_k^\star) - Q_k(x_k, u_k^\star) \leqslant \varepsilon,$$

for all $(x_k, u_k^\star, \tilde{u}_k^\star) \in \mathcal{C}$, thus guaranteeing good quality of the suboptimal control obtained via (7), provided that $x_k$ lies in the projection of $\mathcal{C}$ onto the $x_k$–coordinates.

It is worth noticing that, in several applications, the Q-factor $Q_k(\cdot,\cdot)$ are indeed convex functions. For instance, linear quadratic (LQ) optimal control problems, in which the function $f_k(\cdot,\cdot)$ and $g_k(\cdot,\cdot)$ are defined as

$$f_k(x_k, u_k) = A_k\, x_k + B_k\, u_k, \tag{10a}$$
$$g_k(x_k, u_k) = x_k^\top W_k\, x_k + u_k^\top R_k\, u_k, \tag{10b}$$
$$g_N(x_N) = x_N^\top W_N\, x_N, \tag{10c}$$

where $W_k$, $k = 0, \ldots, N$, are symmetric, positive semi-definite matrices and $R_k$, $k = 0, \ldots, N-1$, are symmetric, positive definite matrices, are characterized by convex Q-factors. In fact, by [28], letting $P_\kappa$ be the solution to

$$\begin{aligned} P_\kappa &= W_\kappa + A_\kappa^\top P_{\kappa+1}\, A_\kappa \\ &\quad - A_\kappa^\top P_{\kappa+1} B_\kappa (R_\kappa + B_\kappa^\top P_{\kappa+1}\, B_\kappa)^{-1} B_\kappa^\top P_{\kappa+1}\, A_\kappa, \end{aligned}$$

starting with $P_N = W_N$, $\kappa = N-1, \ldots, 0$, one has $J_k^\star(x_k) = x_k^\top P_k\, x_k$, whence $Q_k(x_k, u_k) = [\ x_k^\top \quad u_k^\top\ ]\, \Lambda_k\, [\ x_k^\top \quad u_k^\top\ ]^\top$, with

$$\Lambda_k = \begin{bmatrix} W_k + A_k^\top P_{k+1}\, A_k & A_k^\top P_{k+1}\, B_k \\ B_k^\top P_{k+1}\, A_k & R_k + B_k^\top P_{k+1}\, B_k \end{bmatrix}.$$

Therefore, the Q-factor $Q_k(x_k, u_k)$ is convex due to the positive semi-definiteness of the matrix $\Lambda_k$, whose Schur complement [29] equals $P_k$. More generally, a class of problems admitting convex Q-factors are those in which the function $f_k(\cdot,\cdot)$ is defined as in (10a) and the functions $g_0(x_0, u_0), \ldots, g_{N-1}(x_{N-1}, u_{N-1}), g_N(x_N)$ are convex. In fact, in such a case, the function

$$\begin{aligned} Q_{N-1}(x_{N-1}, u_{N-1}) &= g_{N-1}(x_{N-1}, u_{N-1}) \\ &\quad + g_N(A_{N-1} x_{N-1} + B_{N-1} u_{N-1}) \end{aligned}$$

is the sum of a convex function and of another convex function composed with a linear mapping, whence it is convex [19]. Furthermore, since $J_{N-1}^\star(x_{N-1}) = \min_{u_{N-1}} Q_{N-1}(x_{N-1}, u_{N-1})$, i.e., $J_{N-1}^\star$ is the inf-projection of $Q_{N-1}$, it is a convex function. Thus, by considering the definition of Q-factors in (5) and iterating this reasoning backward, we obtain that the Q-factor $Q_k(x_k, u_k)$, $k = 0, \ldots, N-1$, are convex.

## IV. Numerical experiments

In this section, we validate the proposed technique to perform Q-learning via LSE networks through two numerical experiments. In particular, in Section IV-A, we show that, when dealing with linear quadratic optimization problems, the Q-learning via LSE neural networks perform similarly to the actual optimal control. On the other hand, in Section IV-B, we show that, when dealing with a genuinely nonlinear optimization problem, the Q-learning via LSE neural networks perform better than classical Q-learning with quadratic approximation functions.

### A. LQ optimal control of a mechanical system

Consider the mechanical system depicted in Figure 2.

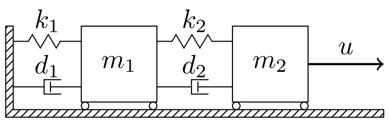

Fig. 2. A simple mechanical system.

The dynamics of this system are given by

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-k_1-k_2}{m_1} & \frac{-d_1-d_2}{m_1} & \frac{k_2}{m_1} & \frac{d_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_2} & \frac{d_2}{m_2} & -\frac{k_2}{m_2} & -\frac{d_2}{m_2} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_2} \end{bmatrix} u,$$

where $x(t) \in \mathbb{R}^4$ denotes the positions and speeds of the bodies having masses $m_1$ and $m_2$. Assuming that $k_1 = 10^3\,\mathrm{N/m}$, $k_2 = 2 \cdot 10^3\,\mathrm{N/m}$, $m_1 = 0.1\,\mathrm{Kg}$, $m_2 = 0.15\,\mathrm{Kg}$, $d_1 = d_2 = 1\,\mathrm{N\,s/m}$, and that the system is controlled through an impulsive force applied to the body having mass $m_2$ every $\tau_M = 1\,\mathrm{s}$ seconds, its discrete-time dynamics are given by a system of the form (10a), with

$$A_k = \begin{bmatrix} 0.0289 & 0.0010 & 0.0475 & 0.0019 \\ -3.0836 & 0.0226 & -6.4323 & 0.0442 \\ 0.0379 & 0.0013 & 0.0621 & 0.0026 \\ -4.1300 & 0.0295 & -8.6020 & 0.0578 \end{bmatrix}, \quad B_k = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 6.6667 \end{bmatrix}.$$

Hence, assume that the objective of the control input $u$ is to minimize the cost function (2), with $N = 8$ and with $g_k$ and $g_N$ given by (10b) and (10c), where $W_k = I$, $k = 0, \ldots, N$, and $R_k = 1$, $k = 0, \ldots, N-1$.

In order to validate the proposed technique, we approximated the Q-factors of the considered optimization problem both via LSE networks ($K = 10$) and via classical polynomial functions. In particular, we first run 100 simulations of the considered mechanical system by selecting the control input $u$ and the initial condition $x_0$ of the plant at random with distribution $\mathcal{N}(1, 0)$ and $\mathcal{N}(I, 0)$, respectively. Then, we approximated the Q-factors both via an LSE networks (by using the technique given in Section III-B) and via second order polynomials in the variables $x_1, x_2, x_3, x_4, u$, whose coefficients have been determined through the least absolute shrinkage and selection operator (LASSO) [30]. Once the LSE networks and the second order polynomials have been determined, we picked other 100 initial conditions $x_0$ of the plant at random with distribution $\mathcal{N}(I, 0)$ and we simulated the behavior of the mechanical system by selecting the input $u$ according to (7), where $U_k(x) = \mathbb{R}$ and the functions $\tilde{Q}_k$ are either the polynomials or the functions in $\mathrm{LSE}_T$ obtained after the first epoch of training. The trajectories of the system obtained in such a way have then been used, together with the ones obtained selecting the control input $u$ at random, to perform a second epoch of training, thus obtaining new polynomials and new $\mathrm{LSE}_T$ functions that approximate the Q-factors $Q_0, \ldots, Q_8$. This procedure has been repeated 6 times, thus obtaining 6 polynomial and 6 $\mathrm{LSE}_T$ approximations of the Q-factors $Q_0, \ldots, Q_8$. Finally, we picked other 100 initial conditions $x_0$ at random with distribution $\mathcal{N}(I, 0)$, we simulated the behavior of the mechanical system by selecting the input $u$ according to (7), with $U_k(x) = \mathbb{R}$ and with $\tilde{Q}_k$ substituted by either the polynomial or the $\mathrm{LSE}_T$ approximation of $Q_k$, and we evaluated the corresponding cost (2). Table I reports the average cost from the application of the pseudo-optimal control laws and the average optimal value obtained using classical LQ tools. Note that, in both the polynomial and $\mathrm{LSE}_T$ cases, the pseudo-optimal inputs have been obtained assuming that the dynamics of the system are not known, whereas the application of the LQ methods required perfect knowledge of both the system and the cost function.

TABLE I

AVERAGE COSTS OBTAINED BY USING THE CONTROL LAW (7).

| Epoch | Polynomials | LSE networks | Optimal solution |
|---|---|---|---|
| 1 | 59.009277 | 59.0255 | 59.009277 |
| 2 | 59.009277 | 59.0112 | 59.009277 |
| 3 | 59.009277 | 59.0127 | 59.009277 |
| 4 | 59.009277 | 59.0171 | 59.009277 |
| 5 | 59.009277 | 59.0125 | 59.009277 |
| 6 | 59.009277 | 59.0126 | 59.009277 |

As shown by such a table, the pseudo-optimal control law obtained by using LSE networks has performance similar to the one obtained by using quadratic polynomials. Note that, since the considered optimal control problem is of the form (10), by the reasoning given in Section III-C, the Q-factors are actually quadratic polynomials and hence the pseudo-optimal control obtained by using polynomial approximates is actually optimal. This implies that the performance obtained by using $\mathrm{LSE}_T$ networks, although pseudo-optimal, is very close to the optimal one.

### *B. Control of lithium ions distribution in the human body*

Lithium ions are one of the most used treatments for the bipolar disorder and for the manic-depressive illness. In [31], a pharmacokinetic model for the distribution of such ions in the human body have been obtained from experimental data. In particular, assuming that the drug is administrated intravenously every 3 hours, the discrete-time dynamics of the drug are given by system (10a) with

$$A_k = \begin{bmatrix} 0.3973 & 0.1025 & 0.3054 \\ 0.7061 & 0.2357 & 0.3545 \\ 0.2607 & 0.0439 & 0.6648 \end{bmatrix}, \quad B_k = \begin{bmatrix} 10.9 \\ 0 \\ 0 \end{bmatrix},$$

where $x = [\, x_1 \;\; x_2 \;\; x_3 \,]^\top$, $x_1$, $x_2$, and $x_3$ denote the plasma, the red blood cells, and the muscle cells concentration of lithium ions, respectively. In order to let the treatment be effective, the concentrations of ion have to satisfy [32]

$$0.4\,\mathrm{nmol/L} \leqslant x_1 \leqslant 0.6\,\mathrm{nmol/L}, \tag{11a}$$

$$0.6\,\mathrm{nmol/L} \leqslant x_2 \leqslant 0.9\,\mathrm{nmol/L}, \tag{11b}$$

$$0.5\,\mathrm{nmol/L} \leqslant x_3 \leqslant 0.8\,\mathrm{nmol/L}. \tag{11c}$$

If the dynamics of the system are known, then the model predictive control technique given in [32] can be used to steer the trajectories of the system to the therapeutic window given in (11). On the other hand, if the dynamics of the system are not known, as it will be assumed in the remainder of this section, it is possible to use Q-learning in order to steer the trajectories of the system to the therapeutic window given in (11). Namely, define the function

$$\begin{aligned} g(x) = {} & \begin{cases} (x_1 - 0.5)^2, & \text{if } 0.4 \leqslant x_1 \leqslant 0.6, \\ |x_1 - 0.5| - 0.0900, & \text{otherwise}, \end{cases} \\ & + \begin{cases} (x_2 - 0.75)^2, & \text{if } 0.6 \leqslant x_2 \leqslant 0.9, \\ |x_2 - 0.75| - 0.1275, & \text{otherwise}, \end{cases} \\ & + \begin{cases} (x_3 - 0.65)^2, & \text{if } 0.5 \leqslant x_3 \leqslant 0.8, \\ |x_1 - 0.65| - 0.1275, & \text{otherwise}, \end{cases} \end{aligned}$$

which is minimized if $x$ is in the therapeutic window and consider the cost function (2) with $N = 8$ (corresponding to a time window of 24 hours), $g_k(x_k, u_k) = g(x_k)$, $k = 0, \ldots, 7$, and $g_N(x_N) = g(x_N)$.

The Q-factors corresponding to the optimization problem have been approximated via both quadratic polynomials and $\mathrm{LSE}_T$ functions ($K = 10$). In particular, since the amount of administered dose should not exceed $5.95\,\mathrm{nmol}$, we firstly run 100 simulations by selecting the control input $u$ and the initial condition $x_0$ of the plant at random with uniform distribution over $[0, 1]^3$ and $[0, 5.95]$, respectively. Then, we approximated the Q-factors both via an LSE networks and via second order polynomials in the variables $x_1, x_2, x_3, u$, whose coefficients have been determined through the LASSO. Once the LSE networks and the second order polynomials have been determined, we picked other 100 initial conditions $x_0$ of the plant at random with uniform distribution over $[0, 1]^3$ and we simulated the behavior of the system by selecting the input $u$ according to (7), where $U_k(x) = [0, 5.95]$ and the functions $\tilde{Q}_k$ are either the polynomials or the functions in $\mathrm{LSE}_T$ obtained after the first epoch of training. These trajectories have then been used, together with the ones obtained by selecting the input uniformly at random, to perform a second epoch of training. This procedure has been repeated 6 times, thus obtaining 6 polynomial and 6 $\mathrm{LSE}_T$ approximations of the Q-factors $Q_0, \ldots, Q_8$. Finally, we picked other 100 initial conditions $x_0$ uniformly at random in $[0, 1]^3$, we simulated the system behavior by selecting $u$ according to (7), with $U_k(x) = [0, 5.95]$ and with $\tilde{Q}_k$ substituted by either the polynomial or the $\mathrm{LSE}_T$ approximation of $Q_k$, and we evaluated the cost (2). Table II reports the average cost yield from the application of the pseudo-optimal control laws and from the application of the optimal control inputs obtained by minimizing the cost function (2) through the Matlab function `fmincon` assuming perfect knowledge of the system.

TABLE II
AVERAGE COSTS OBTAINED BY USING THE CONTROL LAW (7).

| Epoch | Polynomials | LSE networks | Optimal solution |
|---|---|---|---|
| 1 | 3.9925 | 3.9925 | 1.04975 |
| 2 | 3.9925 | 2.4126 | 1.04975 |
| 3 | 3.9925 | 2.2527 | 1.04975 |
| 4 | 3.9925 | 1.9026 | 1.04975 |
| 5 | 3.9925 | 1.6736 | 1.04975 |
| 6 | 3.9925 | 1.6430 | 1.04975 |

As shown by such a table, the Q-learning algorithm that uses LSE networks to approximate the Q-factors outperforms the one using second order polynomial approximates. Furthermore, differently from the pseudo-optimal control laws gathered using polynomial approximates, the ones obtained using LSE networks approach, as the number of epochs increases, the solution found assuming perfect knowledge of both the system dynamics and the cost function.

## V. Conclusions

A novel method to perform Q-learning has been proposed. In particular, it has been shown that LSE neural networks can be used to approximate the Q-factors arising from finite-horizon optimal control problems with continuous state space. The key advantage of using these networks is that they synthesize a convex function and hence are optimized efficiently, without requiring approximations in policy space.

As other Q-learning strategies, the proposed approach may suffer of inadequate exploration since the control input is selected as in (7). However, this issue can be mitigated by selecting $u_k$ at random in $U_k(x_k)$ with a small probability.

Future work will deal with the use of different classes of networks for the approximation of the Q-factors, as, e.g., the class of $\mathrm{DLSE}_T$ networks introduced in [33], which are capable of approximating any continuous functions over convex, compact sets while still possessing a specific difference-of-convex-functions form that makes them optimizable via relatively efficient numerical methods [34], and with the use of these networks to solve infinite horizon optimal control problems in a data-driven setting by using a receding horizon approach similar to that used in model predictive control.

## References

[1] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2011.

[2] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2012.

[3] G. W. Swan, *Applications of optimal control theory in biomedicine*. Dekker, 1984.

[4] C. Diaz, F. Ruiz, and D. Patino, "Smart charge of an electric vehicles station: A model predictive control approach," in *2018 IEEE Conference on Control Technology and Applications*, pp. 54–59, 2018.

[5] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *34th IEEE Conference on Decision and Control*, pp. 560–564, 1995.

[6] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Athena scientific, 1996.

[7] D. P. Bertsekas and S. Ioffe, "Temporal differences-based policy iteration and applications in neuro-dynamic programming," Tech. Rep. LIDS-P-2349, Department of Electrical Engineering and Computer Science , MIT, Cambridge, MA, 1996.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena scientific, 2018.

[10] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703. John Wiley & Sons, 2007.

[11] D. P. De Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations research*, vol. 51, no. 6, pp. 850–865, 2003.

[12] G. C. Calafiore, S. Gaubert, and C. Possieri, "Log-sum-exp neural networks and posynomial models for convex and log-log-convex data," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[13] H. van Hasselt, "Reinforcement learning in continuous state and action spaces," in *Reinforcement Learning: State-of-the-Art* (M. Wiering and M. van Otterlo, eds.), pp. 207–251, Springer, 2012.

[14] K. A. Kopotun, D. Leviatan, and I. A. Shevchuk, "Convex polynomial approximation in the uniform norm: conclusion," *Canadian Journal of Mathematics*, vol. 57, no. 6, pp. 1224–1248, 2005.

[15] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. I, II. Athena scientific, 2012.

[16] R. Bellman, *Dynamic programming*. Courier Corporation, 2013.

[17] C. J. C. H. Watkins, *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[18] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine learning*, vol. 16, no. 3, pp. 185–202, 1994.

[19] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[20] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 281–302, 2000.

[21] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1." `http://cvxr.com/cvx`, Mar. 2014.

[22] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control* (V. Blondel, S. Boyd, and H. Kimura, eds.), Lecture Notes in Control and Information Sciences, pp. 95–110, Springer-Verlag Limited, 2008.

[23] K. G. Vamvoudakis, "Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach," *Systems & Control Letters*, vol. 100, pp. 14–20, 2017.

[24] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[25] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, 2013.

[26] L. E. Scales, *Introduction to non-linear optimization*. Springer, 1985.

[27] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade* (G. Montavon, G. B. Orr, and K.-R. Müller, eds.), pp. 9–48, Springer, 2012.

[28] B. D. O. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[29] F. Zhang, *The Schur complement and its applications*. Springer, 2006.

[30] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[31] B. E. Ehrlich, C. Clausen, and J. M. Diamond, "Lithium pharmacokinetics: Single-dose experiments and analysis using a physiological model," *Journal of pharmacokinetics and biopharmaceutics*, vol. 8, no. 5, pp. 439–461, 1980.

[32] P. Sopasakis, P. Patrinos, H. Sarimveis, and A. Bemporad, "Model predictive control for linear impulsive systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 8, pp. 2277–2282, 2015.

[33] G. C. Calafiore, S. Gaubert, and C. Possieri, "A universal approximation result for difference of log-sum-exp neural networks," 2019. arXiv:1905.08503.

[34] H. A. Le Thi and T. P. Dinh, "DC programming and DCA: thirty years of developments," *Mathematical Programming*, vol. 169, no. 1, pp. 5–68, 2018.