Int. J. Appl. Math. Comput. Sci., 2020, Vol. 30, No. 2, 363–375
DOI: 10.34768/amcs-2020-0028

A LINEAR PROGRAMMING METHODOLOGY FOR APPROXIMATE DYNAMIC PROGRAMMING

HENRY DÍAZ [a,*], ANTONIO SALA [a], LEOPOLDO ARMESTO [b]

[a]University Institute of Industrial Control Systems and Computing
Polytechnic University of Valencia
Camino de Vera, s/n, 46022 València, Spain
e-mail: hendia@posgrado.upv.es

[b]Design and Manufacturing Institute
Polytechnic University of Valencia
Camino de Vera, s/n, 46022 València, Spain

The linear programming (LP) approach to solve the Bellman equation in dynamic programming is a well-known option for finite state and input spaces to obtain an exact solution. However, with function approximation or continuous state spaces, refinements are necessary. This paper presents a methodology to make approximate dynamic programming via LP work in practical control applications with continuous state and input spaces. There are some guidelines on data and regressor choices needed to obtain meaningful and well-conditioned value function estimates. The work discusses the introduction of terminal ingredients and computation of lower and upper bounds of the value function. An experimental inverted-pendulum application will be used to illustrate the proposal and carry out a suitable comparative analysis with alternative options in the literature.

Keywords: linear programming, approximate dynamic programming, control applications, neural networks.

1. Introduction

Optimal control is a widely-used strategy in many applications. These problems are stated as minimising an infinite-time cost index or, in some cases, a finite-time approximation of it, the latter being usually solved via the so-called model predictive control (MPC) methods (Allgower and Zheng, 2012). Another option is iterative LQR setups (Armesto *et al.*, 2015), which may be used to approach the optimal solution by successive linearisations. Additionally, convex optimisation (linear matrix inequalities, LMIs) can be used to obtain upper-cost bounds in some nonlinear systems (Ariño *et al.*, 2017), using Takagi–Sugeno models (Robles *et al.*, 2019).

At the core of many of these techniques lies dynamic programming (DP) (Bertsekas, 2017; Lewis and Liu, 2013). DP solutions can either be off-line model-based ones or on-line ones (with or without a model), giving rise to *adaptive* DP/reinforcement learning paradigms (Liu *et al.*, 2017; Sutton and Barto, 2018). DP estimates a value function $V(x)$ associated with each optimal control problem. Policy iteration (PI) and value iteration (VI) are popular paradigms (Lewis and Liu, 2013) to estimate $V(x)$. PI and VI converge to the optimal solution under mild contraction mapping conditions (Busoniu *et al.*, 2010). However, such mild conditions are actually so only in systems with a finite number of states and actions. DP techniques applied to continuous states are commonly known as *approximate* dynamic programming (ADP), because of the need of using a function approximator such as lookup tables (Busoniu *et al.*, 2010), neural networks (Munos *et al.*, 1999; Díaz *et al.*, 2019) or spline regressors (Marsh and Cormier, 2001; Cervellera *et al.*, 2007), among others.

Other approaches to address optimal control problems include data-based learning strategies such as policy search (Deisenroth *et al.*, 2013), iterative learning control (Tan *et al.*, 2007; Preitl *et al.*, 2007), or learning by demonstration (Armesto *et al.*, 2018). Additionally,

*Corresponding author

the efficiency of RL algorithms may be influenced by using epoch or active exploration methods in order to update the policy (Zajdel, 2013; Zhao *et al.*, 2019); however, these are out of the scope of this paper, so details on them are omitted.

This work will focus on off-line ADP (assuming a batch of data to be available *a priori*). In a generic ADP case, function approximation needs to be used (Busoniu *et al.*, 2010; Powell, 2011; Bertsekas, 2019); however, with arbitrary parametrisations $V(x, \theta)$, PI and VI might not converge. This is also the case for some actor–critic schemes (Lewis and Liu, 2013). A way to avoid PI/VI divergence is trying to minimise the so-called mean-square Bellman error via gradient methods (Munos *et al.*, 1999) or other monotonic optimisation techniques. However, such methods open up the possibility of getting caught on local minima if they are not properly initialised. In our other work (Díaz *et al.*, 2020), we propose LMI-based solutions as a starting point for Bellman error approaches.

As an alternative, De Farias and Van Roy (2003) proposed a linear programming (LP) approach changing equalities to inequalities in the Bellman equation, to generate a lower bound of the value function.

The objective of this work is to propose a methodology, derived from the ideas of De Farias and Van Roy (2003), to approximately solve deterministic optimal control problems with approximate dynamic programming and LP (coined as ADP-LP) on a grid of samples of state and action data points (as a continuous state and input space is, in general, intractable). However, in a continuous-state problem, the data and regressor choices for the approximator $V(x, \theta)$ may end up yielding unbounded solutions or ill-conditioned ones. Our prior work (Díaz *et al.*, 2019) discussed preliminary aspects of this proposal. Here improvements are presented with enhanced discussion, the addition of terminal ingredients and upper bounds to the value function, and experimental validation of the resulting controller.

Specifically, the contributions of this paper are (a) the introduction of two terminal sets: an outer one to penalise exiting the region with data availability and an inner terminal set to avoid anomalous behaviour (such as offset) close to the origin due to the approximation error and control action gridding; (b) the analysis of the relationship with the fuzzy Q-iteration using look-up-tables (LUTs) proposed by Busoniu *et al.* (2010), showing that ADP-LP can avoid the value iteration yielding identical results; and (c) the computation of the upper bound of the value function to determine a gap with respect to the lower bound in order to check whether the regressors are suitable.

In order to check the validity of the approach in practice, an experimental setup with swing-up and stabilisation of an inverted pendulum is presented.

The structure of the paper is as follows. The next section will discuss preliminary definitions and basic concepts. Section 2.3 will define the problem. Section 3 will discuss and define all relevant ingredients of the problem setup. A discussion on approximation quality as well as computational and implementation issues will be given in Section 4. Section 5 will present examples and experiments validating the proposal. Finally, the conclusions will be given in Section 6

2. Preliminaries and the problem statement

Let us consider the discrete dynamic system

$$x_+ = f(x, u), \tag{1}$$

where $x \in \mathbb{X}$ is the state space, $u \in \mathbb{U}$ is the input and $x_+ \in \mathbb{X}$ denotes the successor state so that $f : \mathbb{X} \times \mathbb{U} \mapsto \mathbb{X}$, and $\mathbb{U}$ denotes the set of valid control actions.

A policy $u = \pi(x)$ is a function $\mathbb{X} \to \mathbb{U}$ that represents the closed-loop controller of the system achieving the dynamics $x_+ = f(x, \pi(x))$.

The cost $V_\pi : \mathbb{X} \mapsto \mathbb{R}$, associated with a policy $\pi(x)$ starting from an initial state x_0 will be defined as

$$V_\pi(x_0) := \sum_{k=0}^{\infty} \gamma^k L(x_k, \pi(x_k)) \tag{2}$$

x_k of being the state at time instant k, $L(x, u)$ is a scalar function $\mathbb{X} \times \mathbb{U} \to \mathbb{R}$, also known as "immediate cost" and $0 < \gamma < 1$ a discount factor. In the sequel, we will assume $L(x, u) \geq 0$ for all $(x, u) \in \mathbb{X} \times \mathbb{U}$.

The aim of optimal control is to find an optimal policy $\pi^*(x)$ minimizing $V_\pi(x)$. For an optimal policy to exist and have a finite value function, some assumptions regarding stabilizability and positive-definiteness of L need to hold, which are well studied in a linear case (Lewis *et al.*, 2012). In a general case, directly using (2) in the optimisation problem is intractable, as it involves summing an infinite number of terms. In order to provide a tractable approach, it is well-known that the cost of an arbitrary policy must satisfy the Bellman equation (Bertsekas, 2017):

$$V_\pi(x) = L(x, \pi(x)) + \gamma V_\pi(f(x, \pi(x))), \tag{3}$$

and the value function of the optimal policy, denoted as $V^*(x)$, satisfies

$$V^*(x) = TV^*(x), \tag{4}$$

where the Bellman operator T is defined as

$$TV(x) := \min_{u \in \mathbb{U}} \left(L(x, u) + \gamma V(f(x, u)) \right) \tag{5}$$

Now, from $V^*(x)$, the optimal policy can be obtained,

$$\pi^*(x) = \arg\min_{u \in \mathbb{U}} \left(L(x, u) + \gamma V^*(f(x, u)) \right). \tag{6}$$

2.1. Approximate dynamic programming. Consider a function approximator $V(x,\theta)$, where θ is a vector of adjustable parameters. Based on Bellman's equation (3), given a policy $\pi(x)$, if we define the Bellman residual (Lagoudakis and Parr, 2003) as

$$\begin{aligned}\epsilon_\pi(x,\theta) :=& V_\pi(x,\theta) - L(x,\pi(x)) \\ &- \gamma V_\pi(f(x,\pi(x)),\theta)\end{aligned} \tag{7}$$

then the approximate least-squares solution to the "policy evaluation" problem, given policy $\pi(x)$, is

$$\theta_\pi^* := \arg\min_\theta \|\epsilon_\pi(x,\theta)\|^2, \tag{8}$$

where $\|\epsilon_\pi(x,\theta)\|^2 := \int_{\mathbb{X}} \epsilon_\pi(x,\theta)^2\,\mathrm{d}x$. The integral should be understood as a sum in discrete state spaces; also, in continuous state-spaces, to avoid numerical integration steps, the integral is often understood as the sum of $\epsilon(x_k,\theta)^2$ over a set of fitting points $\{x_1,\dots,x_N\}$.

The motivation for the above optimisation problem (8) lies in the fact that, if $\epsilon_\pi = 0$, then (3) is fulfilled so we have successfully computed the value function of the policy.

Consider now a parametrisation of V_π linear in θ, i.e.,

$$V_\pi(x,\theta) = \phi^T(x)\theta, \tag{9}$$

where the elements of vector $\phi(\cdot)$ are called *regressors*.

With parametrisation (9), the above minimisation (8) is a least-squares problem because

$$\epsilon_\pi(x,\theta) = \left(\phi(x) - \gamma\phi(f(x,\pi(x)))^T\right)\theta - L(x,\pi(x)). \tag{10}$$

Other nonlinear parametrisations might need computing gradients to carry out the minimisation in (8), with the risk of being trapped in local minima.

Analogously, given the value function $V_\pi(x,\theta_\pi^*)$, we can obtain an approximate *policy improvement* by solving

$$\pi_+(x,\theta_\pi^*) := \arg\min_{u\in\mathbb{U}} \left(L(x,u) + \gamma V_\pi(f(x,u),\theta_\pi^*)\right). \tag{11}$$

The (fitted) policy iteration (PI) and (fitted) value iteration (VI) are two popular algorithms to solve the ADP problem. They execute iteratively the *policy evaluation* and *policy improvement* steps until they converge to the optimal policy under special conditions (Lewis and Vrabie, 2009; Lagoudakis and Parr, 2003; Munos and Szepesvári, 2008).

PI/VI convergence. It is well-known that PI and VI algorithms, in their discrete implementation (non-approximated), where the parameters are indeed implemented with tables (Sutton and Barto, 2018), converge to the optimal value function and policy. Also, it is well-known that, in a general case (the approximated version, with continuous states), convergence is not guaranteed anymore. Only in very particular scenarios, where contractivity of the composition of the Bellman operator T plus projection onto the regressor's column space is satisfied, can we guarantee convergence to a given "fixed point" (Busoniu *et al.*, 2010). The main issue is that proving contractivity for a given functional approximator $V(x,\theta)$ is difficult or even impossible because theorems are not usually constructive.

Contractivity can be proved only in very particular cases, such as using a look-up-table with as many regressors as available data (see Busoniu *et al.*, 2010). In practice, this requires a regular discretisation of the state space, which easily falls into the *curse of dimensionality* for high-dimensional systems. It is also obvious that a large number of regressors can represent a drawback in many real scenarios. As an alternative option to avoid convergence issues, monotonic gradient-descent minimisation of the residual (7) may be pursued; however, a good initialisation to avoid local minima, as well as an efficient computation of the gradient of (11), is needed (Díaz *et al.*, 2020).

2.2. Linear programming approaches. Linear programming can be used to get solutions for *exact* DP in discrete state/input spaces (Manne, 1960). In approximate programming, to avoid PI/VI convergence problems, De Farias and Van Roy (2003) propose a lower bound of the value function that satisfies

$$V(x,\theta) \le L(x,u) + \gamma V(f(x,u),\theta), \quad \forall (x,u) \in \mathbb{X}\times\mathbb{U}. \tag{12}$$

Obviously, (12) is equivalent to

$$V(x,\theta) \le \min_{u\in\mathbb{U}} \left(L(x,u) + \gamma V(f(x,u),\theta)\right). \tag{13}$$

i.e., $V(x,\theta) \le TV(x,\theta)$ for all $x \in \mathbb{X}$, which is consistent with (4), but replacing equalities with inequalities. As $\mathbb{X}\times\mathbb{U}$ is finite, we can write (12) as a finite set of inequalities.

Any $V(x,\theta)$ fulfilling (12) is a lower bound of the optimal cost function, i.e., $V(x,\theta) \le V^*(x)$ in the case $\gamma < 1$; indeed, T is a monotonic and contractive operator, so $V \le TV \le T^2V \le \cdots \le T^\infty V = V^*$. If the function approximator is linear in parameters, then maximization of a weighted average of V over the state space, subject to (12), is an LP problem, because (12) can be written as

$$\begin{aligned}\left(\phi^T(x) - \gamma\phi^T(f(x,u))\right)\theta \le L(x,u),& \\ \forall (x,u) \in \mathbb{X}\times\mathbb{U}.&\end{aligned} \tag{14}$$

In this way, the solution of the LP problem tries to approach the actual V^* (which can be considered the largest possible lower bound). The reader is referred to the work of De Farias and Van Roy (2003) for further details.

2.3. Problem statement. As previously discussed, PI/VI have convergence issues, which might be difficult to know in advance for a given parametrisation; gradient-descent minimisation of the Bellman residual can be trapped in local minima, so initialisation may be a crucial decision. On the other hand, the LP approach to ADP provides theoretically guaranteed lower bounds for finite state spaces.

This paper will exploit the LP approach. It needs some refinements (Díaz *et al.*, 2019) to provide meaningful and numerically reliable solutions in practice, to be here discussed in depth. Note that an actual application-specific regressor choice is not in the objectives of this paper: we will only provide validation guidelines to check whether the LP problems are well conditioned and numerically reliable for a given choice of regressors.

Terminal ingredients when the trajectories leave the region of interest will also be incorporated into the methodology (otherwise, the infinite-time problem would be ill-defined unless the region of interest is invariant). In addition to this, an upper bound estimate is a useful enhancement to be discussed in this work: with both upper and lower bounds, we can have a reasonable guess on the accuracy of our value function and, consequently, decide whether or not the regressors are appropriate for our application.

When the number of regressors approaches that of data points, our proposal will encompass earlier ones based on interpolative lookup tables; the relationship with them and the PI/VI options used for its adjustment will be, too, subject to scrutiny.

3. ADP via a linear programming setup

In the sequel, we will assume that a dataset $\mathcal{D} \subseteq \mathbb{D} \times \mathbb{U} \times \mathbb{X}$, consisting on N triplets of data,

$$\begin{aligned}\mathcal{D} :=&\{(x_1, u_1, x_{1+}), (x_2, u_2, x_{2+}), \dots, \\ &(x_N, u_N, x_{N+})\},\end{aligned} \tag{15}$$

is available, where $x_{k+} = f(x_k, u_k)$ and the set $\mathbb{D} \subseteq \mathbb{X}$ will be understood as a "region of interest" in the state space with data availability, i.e., assuming that the dataset covers $\mathbb{D}$ "densely enough." We will denote by $\mathbb{T} := \mathbb{X} \sim \mathbb{D}$ the state space region with no data availability.

The elements of the triple $\xi := (x, u, x_+) \in \mathcal{D}$ will be called, from left to right, as 'source state', 'action' and 'successor state'. When speaking about x_+, we will refer to x as its 'predecessor.' A symbolic representation of the sets $\mathbb{X}$, $\mathbb{D}$ and $\mathbb{T}$, jointly with a few data points in $\mathcal{D}$, is shown in Fig. 1. We will assume that $\mathbb{X}$, $\mathbb{T}$ and $\mathbb{D}$ are such that the successor of any source state in $\mathbb{D}$ lies in $\mathbb{X}$.

Note that the dataset can be obtained either from simulation, if a theoretical model of (1) is available, or via experimental data acquisition.

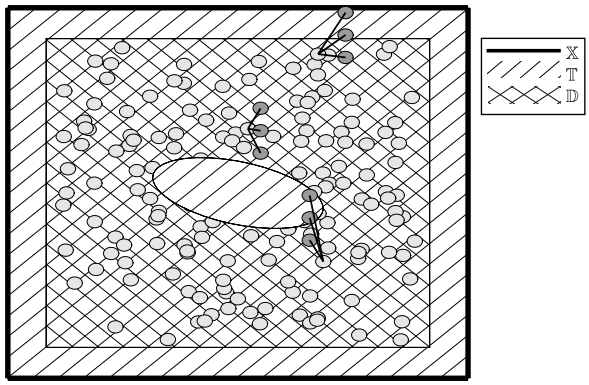


Fig. 1. Illustration of the different sets involved in an optimal control problem ($\mathbb{X} = \mathbb{T} \cup \mathbb{D}$). Light grey circles in the data-availability region $\mathbb{D}$ indicate source states, grey ones indicate successor ones (three cases, to avoid clutter).

3.1. Terminal ingredients. In this paper, the ADP objective is learning an estimate of the value function of the optimal control problem in the region of interest $\mathbb{D}$.

Outside $\mathbb{D}$, we will understand $\mathbb{T}$ as a set of terminal states (terminal set), so the value function $V(x)$ for $x \in \mathbb{T}$ will be known, by assumption, as equal to $V_{\mathbb{T}}(x)$, to be termed as the terminal cost, and *not* approximated by $V(x, \theta)$.

The use of terminal sets is common in model predictive control (MPC) (Allgower and Zheng, 2012); terminal states are also commonplace in discrete dynamic programming problems where a final reward/punishment is received when reaching some absorbing states. These problems end up posing an optimal control problem of reaching $\mathbb{T}$ while minimising

$$V_\pi(x_0) = V_{\mathbb{T}}(x_\tau) + \sum_{k=0}^{\tau-1} \gamma^k \bar{L}(x_k, \pi(x_k)),$$

where final time τ is the reaching time, i.e., $x_k \in \mathbb{D}$ for $k < \tau$ and $x_\tau \in \mathbb{T}$, or infinity if $\mathbb{T}$ is not reached under policy π. With no loss of generality, the terminal cost $V_{\mathbb{T}}(x)$ can be added to the immediate cost when reaching $\mathbb{T}$, i.e., defining

$$\begin{aligned}&L(x, u) \\ &\quad = \begin{cases} \bar{L}(x, u), & x \in \mathbb{D}, f(x, u) \in \mathbb{D}, \\ \bar{L}(x, u) + \gamma V_{\mathbb{T}}(f(x, u)), & x \in \mathbb{D}, f(x, u) \in \mathbb{T}, \\ 0, & x \in \mathbb{T}, \end{cases}\end{aligned} \tag{16}$$

and assuming absorbing terminal states, i.e., $f(x, u) := x$ for $x \in \mathbb{T}$ an we have that $V_\pi(x) \equiv 0$ for all $x \in \mathbb{T}$ for any policy. Thus, with this notation we can express the above reaching control problem as an equivalent infinite-time one with cost L as stated in Section 2.

In a control problem, the "inner" ellipsoidal terminal set in Fig. 1 represents a region where a good performance controller is known (for instance, a linearised LQR one is the standard option in stable linear MPC). The "outer" terminal set would have a "large" terminal cost associated with, say, constraint violations.

3.2. LP constraints and the cost index. With the above-defined dataset and terminal ingredients, constraints (12) will be stated, over the data set, for $k = 1, \dots N$, either as

$$V(x_k, \theta) \leq L(x_k, u_k) + \gamma V(x_{k+}, \theta) \tag{17}$$

if $x_{k+} \in \mathbb{D}$ or, otherwise, if $x_{k+} \in \mathbb{T}$ as

$$V(x_k, \theta) \leq L(x_k, u_k), \tag{18}$$

assuming that the terminal cost has been embedded in L; see (16).

Under the above inequality constraints, the cost index to maximize will be redefined to be

$$M(\theta) = \int_{\mathbb{X}} \vartheta(x) V(x, \theta)\, \mathrm{d}x, \tag{19}$$

where $\vartheta(x)$ is an arbitrary positive weighting to emphasize adjustment in given regions of the state space.

ADP-LP problem. Once regressors $\phi(x)$ for the function approximator are chosen, the cost (19) can be expressed as a linear-in-parameter expression:

$$M(\theta) = \left(\int_{\mathbb{X}} \vartheta(x) \phi^T(x)\, \mathrm{d}x \right) \theta := c^T \theta, \tag{20}$$

where the term between the parentheses is just a vector, once the integral is evaluated. The integral may be considered a "symbolic" expression to be approximated by a finite sum or, alternatively, it can be directly evaluated with numerical or symbolic software for a given choice of $\mathbb{X}$, $\vartheta(\cdot)$ and $\phi(\cdot)$.

Again, if the linear-in-parameter expression of (17)–(18) is written as

$$\left(\phi^T(x_k) - \gamma \phi^T(x_{k+})\right) \theta \leq L(x_k, u_k), \tag{21}$$

if $x_{k+} \in \mathbb{D}$, and

$$\phi^T(x_k) \theta \leq L(x_k, u_k), \tag{22}$$

if $x_{k+} \in \mathbb{T}$, then, from the above discussion, the value function approximation will be carried out by solving the approximate dynamic programming problem of maximising (20) subject to (21)–(22), which, trivially, can be expressed in a compact notation as $A\theta \leq b$ for some matrix A and column vector b.

In the sequel, we will assume that regressors are non-negative, i.e., $\phi(x) \geq 0$ for all $x \in \mathbb{D}$. Without loss of generality, we will assume, too, that $\max_{x \in \mathbb{D}} \phi(x) = 1$, i.e., they are normalised to a maximum value of one over the region $\mathbb{D}$.

Under these assumptions, $\theta = 0$ will always be a feasible solution, because of the assumption that $L(x, u) \geq 0$ for all x and u. Also, the non-negativeness assumption (common in many neural network arrangements) will help understand some of the issues arising due to an incorrect regressor choice, thus avoiding ill-posed LP problems, as discussed next.

3.3. Regressor choice and regularisation. Obviously, regressors should be placed in such a way that they cover the source state data points with high activation. However, in our specific ADP-LP setup, the temporal difference appearing in (21) might produce additional issues if regressors are not carefully arranged,

High successor state activation. If

$$\max_{1 \leq k \leq N} \left(\phi_j(x_k) - \gamma \phi_j(x_{k+})\right) \leq 0, \tag{23}$$

then parameter θ_j in the LP problem will be multiplied by a negative number in all the inequalities. Thus, the ensuing LP problem would end up with an unbounded solution; this should, obviously, be avoided.

The interpretation of (23) is that, for all available data points, the regressor ϕ_j is active with larger intensity on the successor state than on the source state. In order to resolve this issue, a source data point must be added at the point $\hat{x} \in \mathbb{D}$, where the problematic regressor attains a maximum value, i.e., $\phi_j(\hat{x}) = 1$, which exists by assumption. If the data collection phase is finished and that extra data point cannot be gathered, then the peak activation of the regressor should be shifted to, say, the nearest existing data point in $\mathcal{D}$.

Flat regressors, low input excitation. Even if the condition (23) does not occur for a given set of regressors, we might be "close" to it: a possible cause of badly-conditioned setups is having too "flat" regressors. If

$$\max_{1 \leq k \leq N} \left(\phi_j(x_k) - \gamma \phi_j(x_{k+})\right) \leq \beta_1, \tag{24}$$

β_1 being a design parameter (small, positive), then the difference in activation between a state and its successor for regressor ϕ_j is very small, i.e., the regressor takes almost the same value for the source states in the dataset and their successors. Another cause of the problem (24) can be unsuitable input excitation: maybe all tested u are too "small" to move x_k to a successor state sufficiently separated from it to yield significant differences in regressor values.

The result will be undesirably large[1] estimation of parameter θ_j. Decreasing spread parameters in, for instance, Gaussian regressors may solve the problem. If deficient input excitation is suspected, then adding data obtained using larger input excitation is recommended.

3.4. Case of interpolative regressors. This section studies the lookup-table regressor proposals found in, for instance, the work of Busoniu *et al.* (2010), and their relationship to the previously-presented ideas. Let us denote by $\mathcal{D}_x \subseteq \mathbb{D}$ the finite set of source states in the dataset $\mathcal{D}$, with cardinality, say, N_x. Thus, by definition, for any state in $\mathcal{D}_x$ there is a triplet in the dataset $\mathcal{D}$ containing it as a first element, so $\mathcal{D}_x := \{\tilde{x} : \exists \xi = (\tilde{x}, u, x_+) \in \mathcal{D}\}$.

Let us define the matrix $P := [\phi(\tilde{x}_1) \ \dots \ \phi(\tilde{x}_{N_x})]$, x_i being now the enumeration of the N_x elements of $\mathcal{D}_x$ in arbitrary order, with a slight abuse of notation. If P is square (i.e., $M = N_x$, as many regressors as source states) and invertible, then, if we define

$$\widetilde{V} := (V(\tilde{x}_1), \dots, V(\tilde{x}_{N_x}))^T, \tag{25}$$

we can set up the linear, invertible, change of variables given by $\widetilde{V} = P\theta$. Thus, with these regressors we can think of the *value function at source states* $\widetilde{V}$ as being the vector of adjustable parameters. This type of regressors will be denoted as *interpolative* regressors.

The above motivates the use of LUTs as function approximators for ADP-LP, which is, indeed, a sensible option in optimal control applications. This idea has a close relationship with LUT-based options in discrete dynamic programming problems and the fuzzy Q-iteration using LUT found in the work of Busoniu *et al.* (2010) for approximated learning in continuous state spaces. The latter work proposes a standard value-iteration algorithm to adjust the LUT parameters, proving its convergence based on contractive mapping argumentations. Notwithstanding, we will next prove in Section 3.4 that, if contractiveness assumptions hold, the optimal converged value-iteration solution coincides with the one found by solving our ADP-LP problem. Thus, our ADP-LP approach seems, in principle, better than the VI-LUT one, because, apart from providing an identical solution in this LUT case, it can find feasible solutions for alternate regressors even if the contractiveness assumptions fail.

Value iteration under interpolative regressors. In the rest of this section we will assume $\widetilde{V}$ defined in (25) to be the vector of adjustable parameters and $V(x, \tilde{V})$ the function that interpolates between them.

The asynchronous value iteration (Busoniu *et al.*, 2010) is expressed as

$$\widetilde{V}_k^{[l+1]} := T_k V(\tilde{x}_k, \widetilde{V}^{[l]}), \quad k = 1, \dots, N_x, \tag{26}$$

l being the iteration number and operator T_k defined as

$$T_k V(\tilde{x}_k, \widetilde{V}) := \min_{u \in \mathbb{U}(\tilde{x}_k)} \left(L(\tilde{x}_k, u) + \gamma V(f(\tilde{x}_k, u), \widetilde{V}) \right), \tag{27}$$

where $\mathbb{U}(\tilde{x})$ denotes the set of actually tested control actions for the source state $\tilde{x} \in \mathcal{D}_x$ in the given dataset, i.e., $\mathbb{U}(\tilde{x}) := \{u : \exists \xi \ s.t. \ (\tilde{x}, u, \xi) \in \mathcal{D}\}$.

Now, (26) will be shorthanded to

$$\widetilde{V}^{[l+1]} := T_{\mathcal{D}} \widetilde{V}^{[l]} \tag{28}$$

in order to emphasize the application of the dataset-dependent Bellman operator, so its action transforms a value estimate over the source states, $\widetilde{V}^{[l]}$, to another one $\widetilde{V}^{[l+1]}$. Evidently, if the dataset $\mathcal{D}$ were "complete" (i.e., containing all possible transitions in it, in a finite case), then the dataset-dependent operator $T_{\mathcal{D}}$ would actually be the standard Bellman operator.

Obviously, if $T_{\mathcal{D}}$ were contractive, the iterations (28) will converge to a fixed point $\widetilde{V}^* = T_{\mathcal{D}} \widetilde{V}^*$. Now, recall that the ADP-LP constraints, under the $\mathcal{D}$-complete regressors, can be equivalently stated as

$$\widetilde{V} \leq T_{\mathcal{D}} \widetilde{V}. \tag{29}$$

Hence, we can assert that, if $T_{\mathcal{D}}$ is contractive, the fixed point $\widetilde{V}^* = T_{\mathcal{D}} \widetilde{V}^*$ of the value iteration (28) is a feasible solution of the ADP-LP problem posed in Section 3.2. If all coefficients multiplying $\widetilde{V}$ in the cost index $M(\widetilde{V})$ are positive, then the optimal ADP-LP solution coincides with $\widetilde{V}^*$, as maximisation would enforce the decision variables in $\widetilde{V}$ up to equality.

Note that contractiveness of $T_{\mathcal{D}}$, plus positive weights, is a sufficient condition for the ADP-LP solution to obtain in one shot (no iterations) the converged VI solution. But, actually, it is straightforward to realise that, if weight coefficients are positive, under the constraints (29), LP will obtain a bounded fixed point $\tilde{V}^* = T_{\mathcal{D}} \tilde{V}^*$ in all cases in which the LP problem renders feasible and bounded, without the explicit need of contractiveness: our LP proposal succeeds even in situations where the value iteration fails.

In summary, as a conclusion of the section, if VI works with interpolative regressors, so will ADP-LP, yielding exactly the same result; however, ADP-LP will work in other cases, too.

[1] If large parameter values are present, "small" values for the value function estimate $V(x, \theta)$ may be a result of subtraction of large values. Say, if $V(1) = 3 = 5002\phi_1(1) - 5032\phi_2(1)$, then errors in the estimate of any of these parameters would entail large variations in the estimate of $V(x)$. This situation will produce heavy distortion on value function estimates, so it should be avoided.

3.5. Upper bound to the value function. As it stands now, we have computed a sensible lower approximation $V(x,\theta_{LB})$ to the optimal value function. However, it would be desirable to known how "precise" our estimation is with a particular regressor structure. Computation of an "upper" bound to the value function would be, then, helpful: if the gap between upper and lower bounds is large, a denser set of regressors may be advisable.

An option is using the ideas of Rantzer (2006), where relaxed upper and lower bounds are computed via iterations in the form

$$\begin{aligned}\theta_{k+1} =& \min_{\xi} \left(\underline{\nu} L(x,\xi) + \gamma V(f(x,\xi),\theta_{k-1})\right) \\ &\le V(x,\theta_k) \le \overline{\nu} L(x,u) + \gamma V(f(x,u),\theta_k), \\ &\qquad \forall (x,u) \in \mathbb{X}\times\mathbb{U}, \end{aligned} \tag{30}$$

with $\underline{\nu} \le 1 \le \overline{\nu}$ being relaxation parameters, trading off complexity of $V(x,\theta)$ against the gap between $\underline{\nu}$ and $\overline{\nu}$. In this way, (30) embeds (12) in value-iteration settings. The cited works prove that, after k iterations,

$$\min_{\{u_0,\dots,u_k\}} \sum_{i=0}^{k} \gamma^i \underline{\nu} l(x_i,u_i) \le V(x,\theta_k).$$

Thus, if a converged solution is found, denoting its parameters with θ_∞, we have

$$\underline{\nu} V^*(x) \le V(x,\theta_\infty) \le \overline{\nu} V^*(x).$$

However, these iterations may end up yielding infeasible constraints. Convergence or feasibility can only be guaranteed if the approximator precision is very high; see the work of Rantzer (2006, Th. 3) for further details.

As an alternative, we propose to evaluate upper bounds via a linear-programming version of policy iteration, with the standard policy evaluation/improvement steps in an inequality-based version, to be detailed next.

LP policy evaluation (upper bound). Considering an arbitrary policy $\pi(x)$, an LP-based policy evaluation can be stated as

$$L(x,\pi(x)) + \gamma V(f(x,\pi(x)),\theta_\pi) \le V(x,\theta_\pi). \tag{31}$$

Indeed, if a feasible value of θ_π can be found, then $V(x,\theta_\pi) \ge V^*(x)$, by monotonicity and contractiveness of the Bellman operator, proven as in the work of De Farias and Van Roy (2003, Lemma 1), reversing signs. In this case, minimisation of (20), instead of maximisation when computing lower bounds, will be now carried out, to obtain the lowest (on average) feasible upper bound.

These inequalities will always be rendered feasible under the assumptions in Section 3, with $\gamma < 1$. Indeed, in such a setting we can prove that there exists a constant $V_{\max}$ that fulfils (31) as follows. First, note that the successor state $f(x,u)$ can either lie in $\mathbb{D}$ or in $\mathbb{T}$. In each case, $V_{\max}$ should fulfil

$$L(x,u) + \gamma V_{\max} \le V_{\max} \qquad \text{if } f(x,u) \in \mathbb{D}, \tag{32}$$

$$L(x,u) + \gamma V_{\mathbb{T}}(f(x,u)) \le V_{\max} \qquad \text{if } f(x,u) \in \mathbb{T}. \tag{33}$$

Thus, define

$$V_{\mathbb{D},\max} := \frac{1}{1-\gamma} \max_{(x,u)\in\mathbb{D}\times\mathbb{U}} L(x,u),$$

$$V_{\mathbb{T},\max} := \max_{(x,u,f(x,u))\in\mathbb{D}\times\mathbb{U}\times\mathbb{T}} \left(L(x,u) + \gamma V_{\mathbb{T}}(f(x,u))\right),$$

so that we can assert that any constant $V_{\max}$ such that

$$V_{\max} \ge \max(V_{\mathbb{D},\max}, V_{\mathbb{T},\max}) \tag{34}$$

fulfils (31).

As a conclusion, if the chosen regressor arrangement can fit a constant function, the policy evaluation (31) will always be feasible, providing an upper bound of the optimal value function.

Policy improvement. Once a feasible upper bound for the policy evaluation is available, obviously, the new policy arising from (11) will for sure achieve a better upper bound; formally,

$$\begin{aligned} L(x,\pi_+(x)) + \gamma V(f(x,\pi_+(x)),\theta_\pi) & \\ \le L(x,\pi(x)) + \gamma V(f(x,\pi(x)),\theta_\pi) & \\ \le V(x,\theta_\pi). & \end{aligned} \tag{35}$$

LP iteration scheme. Based on the above discussion, we propose, given a starting policy $\pi_0(x)$ and its associated upper bound (31) $V(x,\theta_0)$, to carry out the iterations, starting with $k=0$, of a policy improvement step,

$$\pi_{k+1} := \arg\min_{u\in\mathbb{U}} \left(L(x,u) + \gamma V(f(x,u),\theta_k)\right) \tag{36}$$

and a policy evaluation one,

$$L(x,\pi_{k+1}(x)) + \gamma V(f(x,\pi_{k+1}(x)),\theta_{k+1}) \le V(x,\theta_{k+1}) \tag{37}$$

$$V(x,\theta_{k+1}) \le V(x,\theta_k), \tag{38}$$

where the action of inequality (38) has been added to force an actual "point-wise" improvement of the bound (otherwise, the optimisation index (20) will make the bound decrease "on average", but a point-wise increase cannot be excluded unless $V(x,\theta_{k+1}) \le V(x,\theta_k)$ is, too, enforced).

4. Methodology and a discussion

From the previous considerations, our proposal is to

1. solve the lower-bound ADP-LP setup in Sections 3 and 3.3, obtaining a parameter vector θ_{LB};
2. compute the policy π_0 with (11) from the lower-bound value function estimate;
3. evaluate an upper bound (31) for such a policy, yielding the initial parameter vector θ_0;
4. iterate (37)–(38) and (36) until a suitable stopping criterium is met. Denote the last parameter vector as θ_{UB}.

As discussed in Section 3.4, if we have interpolative and contractive regressors (say, the LUTs in the work of Busoniu *et al.* (2010)), upper and lower bounds in (39) coincide; actually, they are the fixed-point solution of standard value iteration. Our proposal can obtain separate upper and lower bounds in case contractiveness fails and VI does not converge.

Incorporating the above four steps into a general setup involving regressor choice and refinement for practical applications, a flowchart of the methodology presented in this paper is shown in Fig. 2.

As a result of the above steps, we can *approximately* guarantee (see Section 4.1 below for a discussion on the meaning of such approximation)

$$V(x,\theta_{LB}) \le V^*(x) \le V(x,\theta_{UB}), \tag{39}$$

so, if the gap between the upper and lower bounds is deemed excessive for a particular application, a regressor rearrangement (and increasing the number of them, possibly) is recommended. The difference between the associated upper/lower control policies may also be useful to pinpoint regions where an increase in regressor density is recommended.

Note that, nevertheless, the actual choice of new regressors in the case of unsuccessful experimental performance, as well as the analysis of the gap between the upper/lower value functions and the difference in control policies generated by each of them to guide such regressor modification, is out of the scope of this paper, and possibly application-dependent. In summary, the proposals of this paper concentrate on the stages inside the dashed region of the referred flowchart.

4.1. Approximation properties.

As there exist a finite number of data points and regressors, the results of linear programming optimization are only an approximation of the optimal value function. Indeed,

(a) granularity in control actions u makes the computed value function sub-optimal with respect to a continuous space of control;

(b) finite granularity in x forces the fulfilment of the Bellman inequality only at the available data points.

Thus, if we added more data points or more tested control actions at each state to a given dataset, more restrictions on the value function would appear so the obtained result of the LP optimization would be lower: if we denote by $M^*(\mathcal{D},\phi)$ the optimal cost solution of the ADP-LP problem with a dataset $\mathcal{D}$ and regressors ϕ, we have, given two datasets $\mathcal{D}_2 \supseteq \mathcal{D}_1$, then $M^*(\mathcal{D}_2,\phi) \le M^*(\mathcal{D}_1,\phi)$.

Hence, if we define $\overline{\mathcal{D}}$ as the complete dataset containing all elements of $\mathbb{X}\times\mathbb{U}$ and their successors, we can assert $M^*(\overline{\mathcal{D}},\phi) \le M^*(\mathcal{D},\phi)$ for any $\mathcal{D}$.

Now, recalling that the number of regressors is also limited, we can assert that adding more regressors would increase our ADP-LP estimate, i.e., if $\phi^{[2]} \supseteq \phi^{[1]}$, then $M^*(\mathcal{D},\phi^{[1]}) \le M^*(\mathcal{D},\phi^{[2]})$.

If we had the complete dataset $\overline{\mathcal{D}}$ and a regressor set able to fit any arbitrary function over $\mathbb{X}$, we would then be in the non-approximated LP case of the dynamic programming problem. This idealistic regressor will be denoted as $\overline{\Phi}_{\overline{\mathcal{D}}}$. The following diagram can be made:

	Scarce dataset		Complete dataset
Poorer regressor	$M^*(\mathcal{D},\phi^{[1]})$	$\ge$	$M^*(\overline{\mathcal{D}},\phi^{[1]})$
	$\wedge\vert$		$\wedge\vert$
Richer regressor	$M^*(\mathcal{D},\phi^{[2]})$		$M^*(\overline{\mathcal{D}},\overline{\Phi}_{\overline{\mathcal{D}}})$.

In summary, in a finite setting, the bottom-right option would be the exact DP solution and the top-right one would be the proven lower bound proposed by De Farias and Van Roy (2003); on the other hand, in an infinite state/action space, the two options on the right would be elusive unattainable possibilities, to be approximated with a "sufficiently high" number of data triplets and a "sufficiently expressive" set of regressors.

Regarding the upper bound policy evaluation (31), parallel argumentations can be made; so, a diagram similar to the above can be set up, changing the signs of the inequalities. As discussion details are completely analogous, they are omitted.

Implementation issues. Compared with the ordinary least-squares based policy or value iteration in, say, the works of Busoniu *et al.* (2010) or Lewis and Vrabie (2009), our proposal has two distinctive advantages: first, convergence issues are avoided without the need of contractiveness guarantees; second, we provide explicit state-dependent upper and lower bounds of the value function to assess the accuracy of our computations.

The result of the ADP-LP methodology is a value function estimate $V(x,\theta^*)$. As discussed earlier, in order to build the control action, the optimal one is a result of the computation (11). That computation can be carried out on-line if computational resources allow for it, or off-line and stored, say, on a control-map LUT.

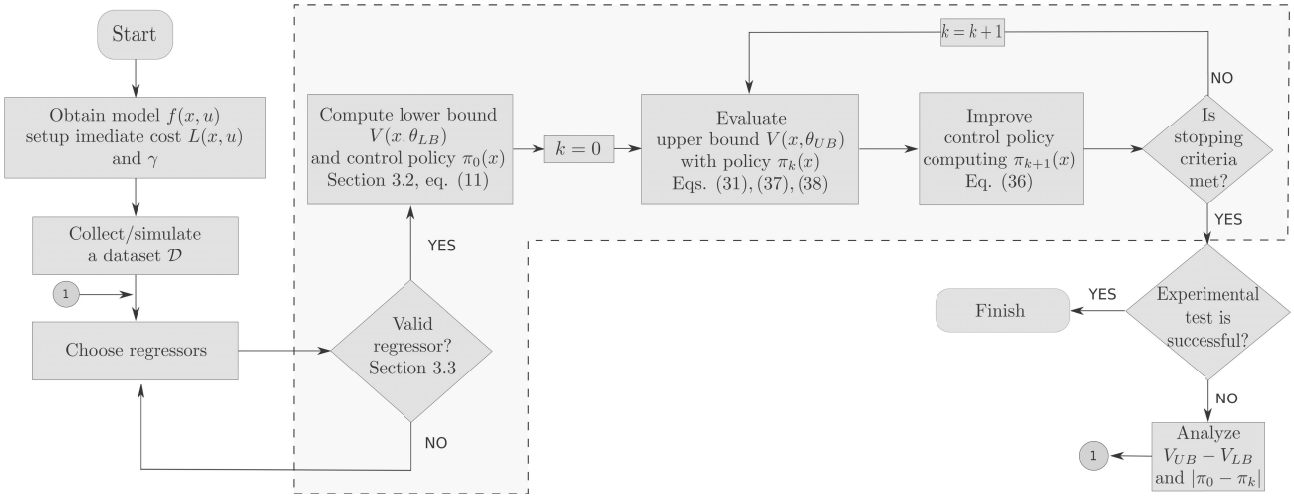


Fig. 2. Flowchart of the proposed methodology.

5. Examples and experimental results

In this section, we show results regarding stabilisation of an inverted pendulum both in simulation and in experimental validation of the resulting controllers.

The goal is stabilising the unstable equilibrium point of an inverted pendulum (see Fig. 3(a)), whose model is

$$\dot{\alpha} = \omega, \tag{40}$$

$$\dot{\omega} = \frac{1}{J}\left(mgl\sin(\alpha) - b\omega + \frac{K}{R}u\right), \tag{41}$$

where α is the pendulum angle, u is the applied voltage, J is the inertia, m is the mass, l is the distance between the rotation axis and the centre of mass, g is the gravity, b is the Coulomb friction, R is the motor electrical resistance and K is the current-torque gain. The available sensor is a Hall-effect encoder measuring angle α. Data acquisition is performed with an NI myRIO-1900 device. Dead-zone compensation was added to compensate for Coulomb friction phenomena in the motor and gears.

The objective is to apply the learnt controller to an actual pendulum prototype shown in Fig. 3(b). Thus, the parameters of the model (40)–(41) will be identified first from the experimental data. Such data are shown in Fig. 4, which consists of a given input profile (bottom plot), generated from two sequences of a swing-up control law (increasing kinetic energy with $u = \text{sign}(\omega)$ plus a random low-pass filtered signal), a PD controller activated when the position is close to the unstable equilibrium, and a disconnection letting the pendulum fall to the bottom stable equilibrium. The control action used in the identification stage is in the interval $[-1, 1]$ V.

The control sampling period is $\Delta t = 0.01$ s. In order to carry out state-feedback control, "measurements" of ω are needed; in this application, a suitably tuned Kalman filter for a double-integrator model was running with tenfold oversampling, and the speed estimate every 10 samples was used for control. Nevertheless, as the only actual measurement is the position one, identification was carried out exclusively with position data. The resulting identified model, Euler discretisation of (40)–(41) is

$$\alpha_+ = \alpha + \Delta t \cdot \omega, \tag{42}$$

$$\omega_+ = \omega + \Delta t \cdot (42.27\sin(\alpha) - 2.27\omega + 24.21u). \tag{43}$$

It achieved a 96.5% fit of the measured position over the validation data segment in Fig. 5 (the command `idnlgrey` from Matlab R2017b System Identification Toolbox® was used).

ADP-LP problem setup. The optimal control problem was set in order have the standard LQR immediate cost,

$$\bar{L}(x,u) = x^T Q x + u^T R u,$$

with

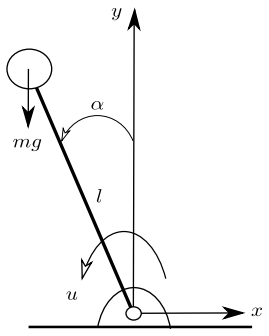
$$Q = \begin{pmatrix} 10 & 0 \\ 0 & 0.1 \end{pmatrix}, \quad R = 1.$$

The discount factor was set to $\gamma := 0.99$.

An ADP-LP problem will be posed and solved based on a dataset $\mathcal{D}$ originating from a uniform gridding 43×43 of the state space $[-\pi,\pi]\times[-15\pi,15\pi]$, and 11-point grid of the control action space $\mathbb{U} := [-1, 1]$. The successor states in $\mathcal{D}$ are generated by simulation of the identified model (42)–(43).

Terminal ingredients. The linearisation of (42)–(43) will be used to obtain a discounted-LQR terminal control law, with an associated quadratic value function

$$V_{LQR}(x) = x^T S x,$$



(a)

(b)

Fig. 3. Inverted pendulum schematic (a) and an actual picture (b).

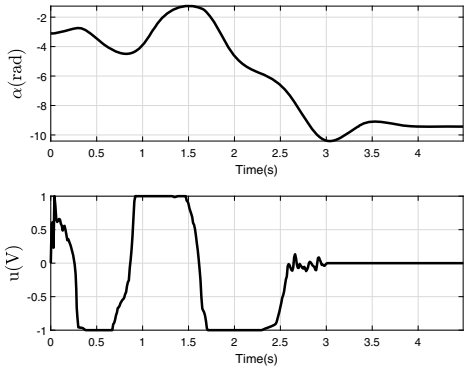


Fig. 4. Test data for identification.

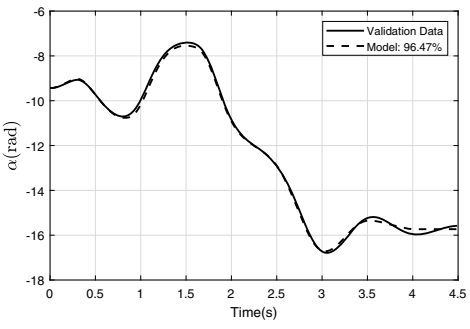


Fig. 5. Validation data and identified model output.

where

$$S = \begin{pmatrix} 265.61 & 19.96 \\ 19.96 & 2.47 \end{pmatrix}$$

is the solution to

$$K = \left(\gamma B^T S B + R\right)^{-1} \gamma B^T S A, \tag{44}$$

$$S = \gamma A^T S \left(A - BK\right) + Q, \tag{45}$$

and the state feedback policy $u = -\left[4.7117 \quad 0.5737\right] x$.

In addition to this, we chose the ellipsoid $\mathbb{T}_1 := \{x : V_{LQR}(x) < 12.10\}$ as an inner terminal region in which we will assume as optimal the discounted LQR solution; this ellipsoidal region is depicted in the latter value function and control map figures, in the centre. Indeed, this region provides a non-saturated LQR control-law. On the other hand, if trajectories exit the gridded state-space region, we will assume that they enter an "outer" terminal region $\mathbb{T}_2 := \{x \in \mathbb{R}^2 : x \notin [-\pi, \pi] \times [-15\pi, 15\pi]\}$ and that an instant penalisation of $V_{\mathbb{T}_2} := 10000$ is considered, basically forcing the system to avoid entering $\mathbb{T}_2$. Thus, the terminal set will be defined as $\mathbb{T} := \mathbb{T}_1 \cup \mathbb{T}_2$, and the terminal cost $V_{\mathbb{T}}(x)$ will be defined as equal to $V_{LQR}(x)$ if $x \in \mathbb{T}_1$, and $V_{\mathbb{T}_2}$ if $x \in \mathbb{T}_2$. With these ingredients, the ADP-LP immediate cost (16) can be built.

Regressor choice. Two regressor choices were tested:

1. a two-dimensional 43×43 LUT (triangular membership functions);
2. a 15×15 uniform grid of RBF neurons, $\phi_j(x) = e^{-(x-\mu_j)^T \Sigma^{-1} (x-\mu_j)}$, normalised to unit sum.

The first regressor, with 1849 adjustable parameters, will be assumed to be close to a "ground-truth" value function, albeit with the error sources discussed in Section 4 due to the finiteness of the grid. However, our objective is testing how the RBF setup with 225 parameters can provide a reasonable value function estimate, in case the 1849-parameter LUT were unattainable due to its memory footprint, as would happen in higher-dimensional problems or with denser grids.

We generated the data gridding and uniformly distributed the RBF centroids so that a neuron centroid coincided with one of each of the three consecutive source states in the dataset. Also, we have removed the regressors (in both the neural network and LUT) whose centroid (interpolation point) belongs to $\mathbb{T}$; otherwise unbounded solutions of the ADP-LP problem would have been obtained. A last decision in the regressor arrangement involved tuning the variance parameter Σ of the RBF neurons so that flat-regressor issues were absent; the finally chosen value was $\Sigma = \mathrm{diag}(0.0542, 12.1847)$. All criteria discussed in Section 3.3 were satisfactorily checked with this regressor proposal. Indeed, we intentionally generated other defective arrangements with

different neuron centroid placement/overlap and neurons inside the inner terminal ellipsoid, and they produced the issues discussed in Section 3.3.

Value function and the control map. The proposed ADP-LP lower bound computations, as well as the subsequent upper bound ones, yielded two estimates $V(x, \theta_{LB})$ and $V(x, \theta_{UB})$. As discussed in Section 3.4, for LUT regressors, both bounds coincide.

We, too, tried standard policy/value iteration setups; the 43×43 LUT was indeed contractive and worked perfectly; see Section 3.4. However, the 15×15 RBF and other RBF and LUT arrangements with a similar number of adjustable parameters failed to converge under the referred widely-used algorithms, illustrating the advantages of the LP approach proposed here.

Figure 6 compares the value function of the 43×43 LUT (the closest we can reasonably get to the ellusive "ground-truth" optimal value function with the given dataset) with the estimates $V(x, \theta_{LB})$ and $V(x, \theta_{UB})$ with our 15×15 RBF. Note that the value function peaks at the lower equilibrium $(-\pi, 0)$, which is intuitively expected. Even if the white surface LUT result was not available, if the gap between the light gray and gray surfaces was deemed excessive, that would advise increasing the flexibility of the regressors with more neurons.

The control map derived from $V(x, \theta_{UB})$ appears in Fig. 7. It resembles swing-up controllers which increase the mechanical energy until it is enough to reach the upper position. Figure 8 depicts the 43×43 LUT control map, showing that the 15×15 RBF achieved a control map that is close to it.

Time simulation and experiments. Figure 9 depicts a representative example of the system trajectories with both simulated (discontinuous line) and actual experiments (continuous line) with the controller arising from the value function $V(x, \theta_{UB})$ estimated by the 15×15 RBF. The simulated pendulum produces a swing-up control law as previously discussed, where we can clearly appreciate that the system commutes to the terminal LQR control law at time instant $t \approx 4$ s. The real pendulum produces a similar response.

A phase plane of the trajectory starting at point $x = (-\pi, 0)$ is shown, together with a trajectory starting at $x = (-\pi, 40)$, in Fig. 10, showing satisfactory experimental behaviour: the mechanical energy is being reduced by braking until the system has the correct amount of energy to enter the LQR terminal region.

6. Conclusions

Based on earlier linear-programming approaches to approximate dynamic programming, this paper has

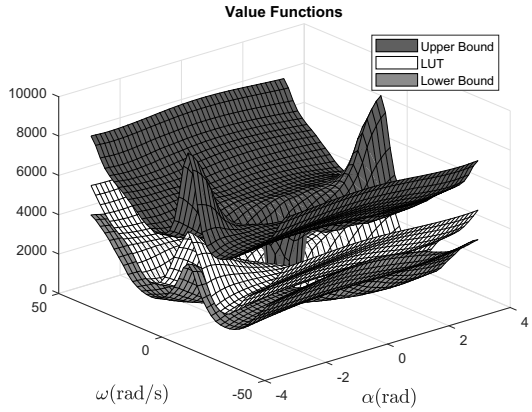


Fig. 6. Value function estimates: 43×43 LUT (white), 15×15 lower bound $V(x, \theta_{LB})$ (light grey), 15×15 upper bound $V(x, \theta_{UB})$ (grey).

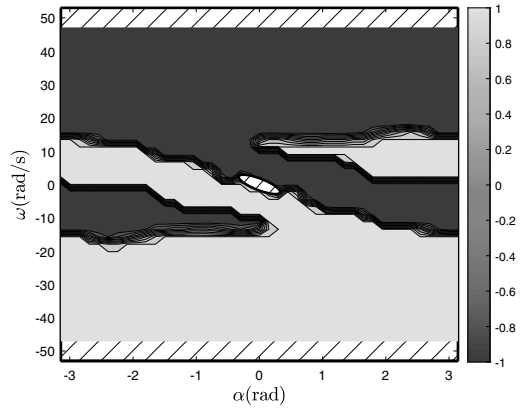


Fig. 7. Control policy, 15×15 RBF neurons using θ_{UB}, plus crossed region $\mathbb{T}$.

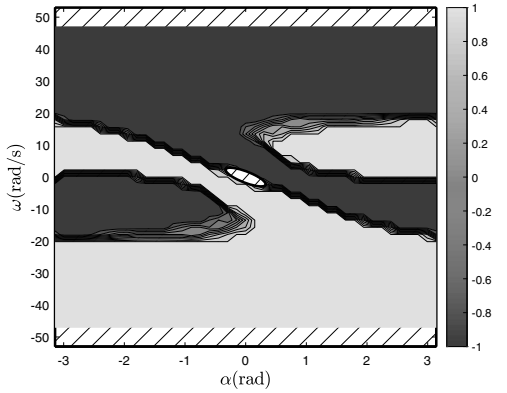


Fig. 8. Control policy, LUT 43×43 (crossed region is $\mathbb{T}$).

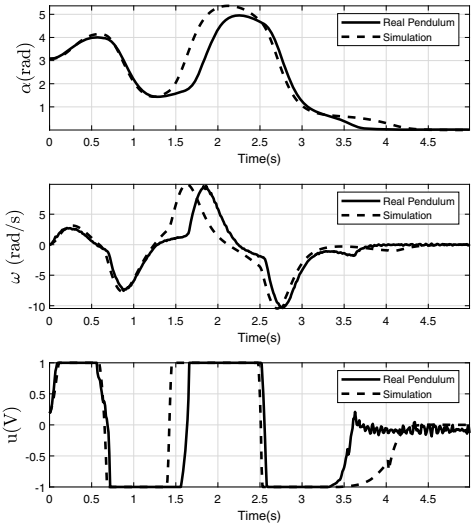


Fig. 9. Simulation and experimental responses with a 15×15 RBF: position, speed and control action (top to bottom).

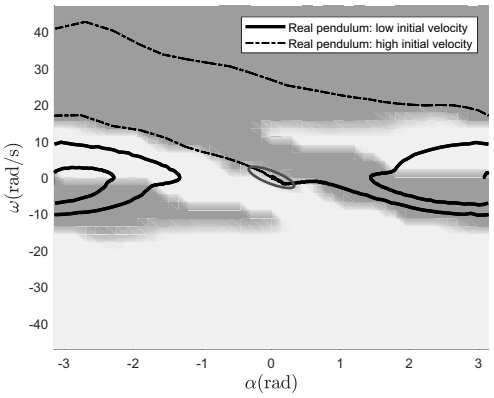


Fig. 10. Phase planes of two trajectories with a 15×15 RBF controller. Background image corresponds to Fig. 7, inner LQR terminal set outlined with its boundary ellipsoid.

presented a methodology to produce well-conditioned and reasonably accurate solutions to optimal control problems. The methodology includes terminal ingredients and the regressor regularisation. The results are lower and upper bounds of the value function, so the gap between them can guide the choice of the regressors . The validity of the approach has been experimentally tested. Even if equally-spaced RBF neurons and LUTs have been used in the examples in this work, other neuron distributions (or problem-specific regressors) may also be employed in order to obtain good controllers with a reduced number of parameters. Future work will address the extension of the approach to Q-function approximation and, additionally, the use of efficient/sparse regressor arrangements in higher-dimensional spaces.

Acknowledgment

The authors are grateful for the financial support of the Spanish Ministry of Economy and the European Union, grant DPI2016-81002-R (AEI/FEDER, UE), and the PhD grant from the Government of Ecuador (SENESCYT).

References

Allgower, F. and Zheng, A. (2012). *Nonlinear Model Predictive Control*, Springer, New York, NY.

Ariño, C., Querol, A. and Sala, A. (2017). Shape-independent model predictive control for Takagi–Sugeno fuzzy systems, *Engineering Applications of Artificial Intelligence* **65**(1): 493–505.

Armesto, L., Girbés, V., Sala, A., Zima, M. and Šmídl, V. (2015). Duality-based nonlinear quadratic control: Application to mobile robot trajectory-following, *IEEE Transactions on Control Systems Technology* **23**(4): 1494–1504.

Armesto, L., Moura, J., Ivan, V., Erden, M.S., Sala, A. and Vijayakumar, S. (2018). Constraint-aware learning of policies by demonstration, *International Journal of Robotics Research* **37**(13–14): 1673–1689.

Bertsekas, D.P. (2017). *Dynamic Programming and Optimal Control*, Vol. 1, 4th Edn, Athena Scientific, Belmont, MA.

Bertsekas, D.P. (2019). *Reinforcement Learning and Optimal Control*, Athena Scientific, Belmont, MA.

Busoniu, L., Babuska, R., De Schutter, B. and Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*, CRC Press, Boca Raton, FL.

Cervellera, C., Wen, A. and Chen, V.C. (2007). Neural network and regression spline value function approximations for stochastic dynamic programming, *Computers & Operations Research* **34**(1): 70–90.

De Farias, D.P. and Van Roy, B. (2003). The linear programming approach to approximate dynamic programming, *Operations Research* **51**(6): 850–865.

Deisenroth, M.P., Neumann, G. and Peters, J. (2013). A survey on policy search for robotics, *Foundations and Trends in Robotics* **2**(1–2): 1–142.

Díaz, H., Armesto, L. and Sala, A. (2019). Metodología de programación dinámica aproximada para control óptimo basada en datos, *Revista Iberoamericana de Automática e Informática industrial* **16**(3): 273–283.

Díaz, H., Armesto, L. and Sala, A. (2020). Fitted Q-function control methodology based on Takagi–Sugeno systems, *IEEE Transactions on Control Systems Technology* **28**(2): 477–488.

Lagoudakis, M.G. and Parr, R. (2003). Least-squares policy iteration, *Journal of Machine Learning Research* **4**(Dec): 1107–1149.

Lewis, F.L. and Liu, D. (2013). *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, Wiley, Hoboken, NJ.

Lewis, F.L. and Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control, *IEEE Circuits and Systems Magazine* **9**(3): 32–50.

Lewis, F., Vrabie, D. and Syrmos, V. (2012). *Optimal Control*, 3rd Edn, John Wiley & Sons, Hoboken, NJ.

Liu, D., Wei, Q., Wang, D., Yang, X. and Li, H. (2017). *Adaptive Dynamic Programming with Applications in Optimal Control*, Springer, Berlin.

Manne, A.S. (1960). Linear programming and sequential decisions, *Management Science* **6**(3): 259–267.

Marsh, L.C. and Cormier, D.R. (2001). *Spline Regression Models*, Number 137, Sage, Thousand Oaks, CA.

Munos, R., Baird, L.C. and Moore, A.W. (1999). Gradient descent approaches to neural-net-based solutions of the Hamilton–Jacobi–Bellman equation, *International Joint Conference on Neural Networks, Washington, DC, USA*, Vol. 3, pp. 2152–2157.

Munos, R. and Szepesvári, C. (2008). Finite-time bounds for fitted value iteration, *Journal of Machine Learning Research* **9**(May): 815–857.

Powell, W.B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd Edn, Wiley, Hoboken, NJ.

Preitl, S., Precup, R.-E., Preitl, Z., Vaivoda, S., Kilyeni, S. and Tar, J.K. (2007). Iterative feedback and learning control. servo systems applications, *IFAC Proceedings Volumes* **40**(8): 16–27.

Rantzer, A. (2006). Relaxed dynamic programming in switching systems, *IEE Proceedings: Control Theory and Applications* **153**(5): 567–574.

Robles, R., Sala, A. and Bernal, M. (2019). Performance-oriented quasi-LPV modeling of nonlinear systems, *International Journal of Robust and Nonlinear Control* **29**(5): 1230–1248.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement Learning: An Introduction*, 2nd Edn, MIT Press, Cambridge, MA.

Tan, K., Zhao, S. and Xu, J. (2007). Online automatic tuning of a proportional integral derivative controller based on an iterative learning control approach, *IET Control Theory Applications* **1**(1): 90–96.

Zajdel, R. (2013). Epoch-incremental reinforcement learning algorithms, *International Journal of Applied Mathematics and Computer Science* **23**(3): 623–635, DOI: 10.2478/amcs-2013-0047.

Zhao, D., Liu, J., Wu, R., Cheng, D. and Tang, X. (2019). An active exploration method for data efficient reinforcement learning, *International Journal of Applied Mathematics and Computer Science* **29**(2): 351–362, DOI: 10.2478/amcs-2019-0026.

Henry Díaz received his BSc degree in electronics and control engineering from the National Polytechnic School, Quito, Ecuador, in 2011, his MSc degree in automation and industrial informatics in 2015 and his PhD degree in automation, robotics and industrial computing in 2020, both from the Polytechnic University of Valencia, Spain. His current research interests include reinforcement learning, approximate dynamic programming, control systems, and robotics.

Antonio Sala was born in 1968. He received his MSc degree in electrical engineering and his PhD degree in control engineering from the Polytechnic University of Valencia (UPV), Spain, in 1993 and 1998, respectively. He has been the chair professor with the Systems and Control Engineering Department, UPV, since 2009. He has published over 75 journal papers and more than 140 conference ones. Professor Sala has served as an associate editor of *IEEE Transactions on Fuzzy Systems, Fuzzy Sets and Systems* and *Revista Iberoamericana de Automática e Informática Industrial*.

Leopoldo Armesto received his BSc degree in electronic engineering, his MSc degree in control systems engineering, and his PhD degree in automation and industrial computer science from Universitat Politécnica de Valéncia (UPV), Spain, in 1998, 2001, and 2005, respectively, where he had held a PhD scholarship for three years at the Department of Systems and Control Engineering Department (DISA). Since 2004 and 2016, respectively, he has been an assistant professor and a senior lecturer at DISA, UPV. He is also a member of the Robotics and Automation Research Group of the Design and Manufacturing Institute at UPV. He is the author or a coauthor of 16 journal and more than 70 conference papers. His current research interests include mobile robotics, optimal control, advanced driving assistance systems, educational robotics, 3-D printing and reinforcement learning.

Received: 26 October 2019
Revised: 27 February 2020
Accepted: 2 March 2020