Received: 29 October 2019 | Revised: 9 February 2020 | Accepted: 12 February 2020
DOI: 10.1002/acs.3106

RESEARCH ARTICLE

WILEY

Adaptive dynamic programming for model-free tracking of trajectories with time-varying parameters

Florian Köpf[1] | Simon Ramsteiner[1] | Luca Puccetti[1,2] | Michael Flad[1] | Sören Hohmann[1]

[1]Institute of Control Systems, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

[2]BMW Group, Munich, Germany

Correspondence
Florian Köpf, Institute of Control Systems, Karlsruhe Institute of Technology (KIT), Kaiserstr. 12, 76131 Karlsruhe, Germany.
Email: florian.koepf@kit.edu

Summary
Recently proposed adaptive dynamic programming (ADP) tracking controllers assume that the reference trajectory follows time-invariant exo-system dynamics—an assumption that does not hold for many applications. In order to overcome this limitation, we propose a new Q-function that explicitly incorporates a parametrized approximation of the reference trajectory. This allows learning to track a general class of trajectories by means of ADP. Once our Q-function has been learned, the associated controller handles time-varying reference trajectories without the need for further training and independent of exo-system dynamics. After proposing this general model-free off-policy tracking method, we provide an analysis of the important special case of linear quadratic tracking. An example demonstrates that our new method successfully learns the optimal tracking controller and outperforms existing approaches in terms of tracking error and cost.

KEYWORDS
adaptive control, adaptive dynamic programming, intelligent control, learning systems

1 | INTRODUCTION

Adaptive and iterative learning controllers are a powerful tool in case of unknown or partially unknown system dynamics[1-5] or in multiagent coordination problems.[6] For the data-based tuning of optimal controllers, where the objective is to minimize a cost functional, adaptive dynamic programming (ADP), which is a method of reinforcement learning, has recently gained extensive attention.[7] In ADP, the controller adapts its behavior based on its interaction with an unknown system and the associated cost signals.[8]

In order to employ ADP in a broader range of applications, this work focuses on the ADP tracking case. Here, the aim is to track a desired reference trajectory optimally w.r.t. a given objective function for a system with unknown dynamics and where no explicit system model is used (ie, the model-free setting is considered). The objective function quantifies the control objectives and typically penalizes the control effort and/or the deviation of the system state from the desired trajectory. Examples that require the tracking of flexible and time-varying trajectories are the longitudinal

A preliminary version of this manuscript is available at arXiv.

control of automated vehicles with desired velocity profiles,[9] lateral steering controllers with lane-following objectives,[10] or various control problems in process engineering.[11] In the model-free case, the trajectory-tracking optimization problem is a challenging learning task, because the long-term cost, ie, the value of a state, changes depending on the reference trajectory. Consequently, a controller that has learned to solve a regulation problem cannot be transferred to the tracking case directly. In order to improve the tracking capabilities, the course of the reference trajectory should be considered.

In the literature, there are several ADP tracking approaches in discrete time[12-17] and continuous time.[18-20] For the tracking control of a simulated autonomous underwater vehicle, Shi et al[12] extend the system state by the current and next reference value and perform pseudo averaged Q-learning. While this is suited for their specific application, the reference trajectory has a very limited preview and the system state can lag behind in general as the controller reacts to current deviations rather than considering the course of the trajectory. Ng et al[13] use reinforcement learning to learn maneuvers for autonomous helicopter flight. Their neural network controller uses the deviation from the desired position as input. Thus, in order to follow a desired trajectory, steady-state goal positions are set consecutively while using a projection of the trajectory in order to reduce the effect of lagging behind. However, the controller is never aware of the course of the trajectory but is only provided a goal position. Furthermore, working only with the deviation from the desired position is only valid for systems that are invariant with respect to the current absolute position. This is approximately valid for the helicopter position but is violated in other instances, for example, in a spring-mass-damper system.

A different, widely used ADP tracking approach is to assume that the reference trajectory $\boldsymbol{r}_k$ can be modeled by means of a time-invariant exo-system $\boldsymbol{r}_{k+1} = \boldsymbol{f}_{\text{ref}}(\boldsymbol{r}_k)$ (or $\dot{\boldsymbol{r}}(t) = \boldsymbol{f}_{\text{ref}}(\boldsymbol{r}(t))$ in the continuous case).[14-20] Then, an approximated value function (or Q-function) is learned, which rates different states (or state-action combinations) w.r.t their expected long-term cost considering the reference dynamics $\boldsymbol{f}_{\text{ref}}$. With this value function, approximated optimal control laws are derived, either directly due to analytical relations between the value function and the optimal controller[14-16,18,20] or by tuning an actor neural network based on the learned value function.[17,19] However, a limitation of this ADP tracking approach is that whenever the reference trajectory and thus the function $\boldsymbol{f}_{\text{ref}}$ change, the learned value function and the learned controller are not valid anymore and need to be retrained. Consequently, the exo-system tracking case with time-invariant reference dynamics $\boldsymbol{f}_{\text{ref}}$ is not suited for the above-mentioned applications, which require tracking of flexible and time-varying trajectories.[21]

Another possible approach would be to learn different value functions for each exo-system $\boldsymbol{f}_{\text{ref}}$ and automatically switch between them. This idea is inspired by the multiple-model approach presented by Kiumarsi et al,[22] which is designed to cope with time-varying system dynamics. Here, self-organizing maps are used to determine the contribution of each sub-model to the value function. However, when transferring this idea to time-varying exo-system dynamics, new submodels have to be trained for each exo-system $\boldsymbol{f}_{\text{ref}}$. Furthermore, because the training is performed on-policy, new data have to be collected during each submodel training and the data cannot be reused in contrast to using off-policy methods. Finally, the multiple-model method[22] allows only partially-unknown system dynamics. While the drift dynamics $f(x)$ can be unknown, the input dynamics $g(x)$ have to be known. In comparison, our method does not require the training of submodels, is off-policy, and works completely model-free.

In contrast to existing methods, our idea is to define a state-action-*reference* Q-function that explicitly incorporates the course of the reference trajectory in contrast to the commonly used Q-function (see, eg, Sutton and Barto[23]) that only depends on the current state $\boldsymbol{x}_k$ and control $\boldsymbol{u}_k$. This general idea has first been proposed in our previous work,[24] where the reference $\boldsymbol{r}_k$ is given on a finite horizon and assumed to be zero thereafter. Thus, the number of weights to be learned depends on the horizon on which the reference trajectory is considered. As the reference trajectory is given for each time step, this allows high flexibility, but the sampling time and (unknown) system dynamics significantly influence the reasonable horizon length and thus the number of weights to be learned.

Based on the above-mentioned challenges, our major idea and contribution in the present work are to approximate the reference trajectory at each time step k by means of a corresponding parameter set $\boldsymbol{P}_k$, in order to compress the information about the reference compared to our previous work[24] and incorporate this parameter set into a new Q-function. In doing so, the Q-function explicitly represents the dependency of the expected long-term cost on the desired reference trajectory. The associated optimal controller is able to cope with time-varying parametrized references. We term this method *parametrized reference ADP (PRADP)*. Our proposed model-free ADP tracking approach allows reference trajectories that are more flexible than trajectories generated by time-invariant exo-systems.[14-20] In contrast to including only the desired state (or its deviation) into the Q-function and hence into the ADP controller, we consider the course of the trajectory.

This prevents the system state from lagging behind and allows the controller to thoroughly understand the current tracking situation. Finally, no retraining of the controller is required because the learned Q-function explicitly depends on the parameter $\boldsymbol{P}_k$, which parametrizes the reference trajectory and its future course from time step k on. Consequently, the optimal controller, which is derived from the learned Q-function, is a function of both the state $\boldsymbol{x}_k$ and the approximated course of the reference trajectory by means of $\boldsymbol{P}_k$.[25]

Our main contributions are summarized as follows.

- The introduction of a new reference-dependent Q-function that explicitly depends on the reference parameter $\boldsymbol{P}_k$, where $\boldsymbol{P}_k$ can be arbitrary and time-varying. This Q-function is a more generalized tracking approach compared to the state-of-the-art ADP tracking controllers and does not require to be retrained when the reference trajectory changes.
- Function approximation of this Q-function in order to realize temporal difference (TD) learning (cf. Sutton[26]).
- Rigorous analysis of the form of this Q-function and its associated optimal control law in the special case of linear-quadratic (LQ) tracking. We show that under reasonable assumptions, existence and uniqueness of the optimal control law can be guaranteed.
- A comparison of our proposed method with algorithms assuming a time-invariant exo-system $\boldsymbol{f}_{\text{ref}}$ and the ground truth optimal tracking controller.

In the next section, the general problem definition is given. Then, PRADP is proposed in Section 3. Simulation results and a discussion are given in Section 4 before the article is concluded.

2 | GENERAL PROBLEM DEFINITION

Consider a discrete-time controllable system

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

where $k \in \mathbb{N}_0$ is the discrete time step, $\boldsymbol{x}_k \in \mathbb{R}^n$ is the system state, and $\boldsymbol{u}_k \in \mathbb{R}^m$ is the control input. The system dynamics $\boldsymbol{f}(\cdot)$ is assumed to be *unknown*. Furthermore, let a parametrized reference trajectory $\boldsymbol{r}(\boldsymbol{P}_k, i) \in \mathbb{R}^n$ be described by

$$\boldsymbol{r}(\boldsymbol{P}_k, i) = \boldsymbol{P}_k \boldsymbol{\rho}(i) = \begin{bmatrix} \boldsymbol{p}_{k,1}^{\mathsf{T}} \\ \boldsymbol{p}_{k,2}^{\mathsf{T}} \\ \vdots \\ \boldsymbol{p}_{k,n}^{\mathsf{T}} \end{bmatrix} \boldsymbol{\rho}(i). \tag{2}$$

Thus, at any time step k, the reference trajectory is parametrized by means of $\boldsymbol{P}_k \in \mathbb{R}^{n \times p}$ and given basis functions $\boldsymbol{\rho}(i) \in \mathbb{R}^p$. Here, $i \in \mathbb{N}_0$ denotes the time step on the reference from the local perspective at time k. Thus, for $i = 0$, the reference at time step k results and $i > 0$ yields a prediction of the reference for future time steps. Therefore, in contrast to methods that assume that the reference follows the time-invariant exo-system dynamics $\boldsymbol{f}_{\text{ref}}$, the parameters $\boldsymbol{P}_k$ in (2) can be time-varying, allowing much more diverse reference trajectories.

The control objective is that the system state $\boldsymbol{x}_{k+i}$ follows the desired reference trajectory $\boldsymbol{r}(\boldsymbol{P}_k, i)$, $i = 0, 1, \dots$ optimally w.r.t. an objective function J_k. Thus, the aim is to obtain a controller through learning, which minimizes the cost

$$J_k = \sum_{i=0}^{\infty} \gamma^i c(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}, \boldsymbol{r}(\boldsymbol{P}_k, i)), \tag{3}$$

for a system with unknown dynamics. Here, $\gamma \in [0, 1)$ is a discount factor and $c(\cdot)$ denotes a nonnegative single-step cost that can, for example, penalize deviations of the system state $\boldsymbol{x}_{k+i}$ from the desired state $\boldsymbol{r}(\boldsymbol{P}_k, i)$ as well as the control effort. We define our general problem as follows.

Problem 1. For a given parametrization of the reference by means of $\boldsymbol{P}_k$ according to (2), an optimal control sequence that minimizes the cost (3) is denoted by $\boldsymbol{u}_k^*, \boldsymbol{u}_{k+1}^*, \dots$ and the associated cost by J_k^*. The system dynamics is unknown. At each time step k, find $\boldsymbol{u}_k^*$.

3 | PARAMETRIZED REFERENCE ADP

In order to solve Problem 1, we first propose a new, modified Q-function whose minimizing control represents a solution $\boldsymbol{u}_k^*$ to Problem 1. In the next step, we parametrize this Q-function by means of linear function approximation. Then, we apply least-squares policy iteration (LSPI) (cf. Lagoudakis and Parr[27]) in order to learn the unknown Q-function weights from data without requiring a system model. Finally, we discuss the structure of this new Q-function for the LQ tracking problem, where analytical insights are possible.

3.1 | Proposed Q-function

When minimizing the cost J_k as given in (3), the relative position i on the current reference trajectory that is parametrized by means of $\boldsymbol{P}_k$ according to (2) needs to be considered. In order to do so, one could explicitly incorporate the relative time i into the Q-function that is used for ADP. This would yield a Q-function of the form $Q(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k, i)$. However, this would unnecessarily increase the complexity of the Q-function and hence the challenge to approximate and learn such a Q-function. Thus, we decided to implicitly incorporate the relative time i on the current reference trajectory parametrized by $\boldsymbol{P}_k$ into the reference trajectory parametrization. This can be achieved by using a shifted parameter matrix $\boldsymbol{P}_k^{(i)}$ according to the following definition.

Definition 1 (Shifted parameter matrix $\boldsymbol{P}_k^{(i)}$). Let the matrix $\boldsymbol{P}_k^{(i)}$ be defined such that

$$\boldsymbol{r}\left(\boldsymbol{P}_k^{(i)}, j\right) = \boldsymbol{r}(\boldsymbol{P}_k, i+j) \tag{4a}$$

$$\Leftrightarrow \boldsymbol{P}_k^{(i)} \boldsymbol{\rho}(j) = \boldsymbol{P}_k \boldsymbol{\rho}(i+j). \tag{4b}$$

Thus,

$$\boldsymbol{P}_k^{(i)} = \boldsymbol{P}_k \boldsymbol{T}(i), \tag{5}$$

is a modified version of $\boldsymbol{P}_k = \boldsymbol{P}_k^{(0)}$ such that the associated reference trajectory is shifted by i time steps, where $\boldsymbol{T}(i)$ is a suitable matrix. Note that $\boldsymbol{T}(i)$ is in general ambiguous as in the general case $p > 1$, the system of (4b) used to solve for $\boldsymbol{P}_k^{(i)}$ is underdetermined. Thus, $\boldsymbol{T}(i)$ can be any matrix such that (4) holds.

With $\boldsymbol{P}_k^{(i)}$ as in Definition 1, our proposed Q-function that explicitly incorporates the reference trajectory by means of $\boldsymbol{P}_k$ is given as follows.

Definition 2 (Parametrized reference Q-function). Let

$$\begin{aligned} Q^{\boldsymbol{\pi}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k) &= c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0)) + \sum_{i=1}^{\infty} \gamma^i c\left(\boldsymbol{x}_{k+i}, \boldsymbol{\pi}\left(\boldsymbol{x}_{k+i}, \boldsymbol{P}_k^{(i)}\right), r(\boldsymbol{P}_k, i)\right) \\ &= c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0)) + \gamma Q^{\boldsymbol{\pi}}\left(\boldsymbol{x}_{k+1}, \boldsymbol{\pi}\left(\boldsymbol{x}_{k+1}, \boldsymbol{P}_k^{(1)}\right), \boldsymbol{P}_k^{(1)}\right). \end{aligned} \tag{6}$$

Here, $\boldsymbol{\pi} : \mathbb{R}^n \times \mathbb{R}^{n \times p} \to \mathbb{R}^m$ denotes the current control policy. With this definition, $Q^{\boldsymbol{\pi}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)$ represents the accumulated discounted cost if the system is in state $\boldsymbol{x}_k$, the control $\boldsymbol{u}_k$ is applied at time k, and the policy $\boldsymbol{\pi}(\cdot)$ is followed thereafter and the reference trajectory is parametrized by $\boldsymbol{P}_k$. Based on (6), the optimal Q-function $Q^*(\cdot)$ is given by

$$\begin{aligned} Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k) &= c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0)) + \min_{\boldsymbol{\pi}} \gamma Q^{\boldsymbol{\pi}}\left(\boldsymbol{x}_{k+1}, \boldsymbol{\pi}\left(\boldsymbol{x}_{k+1}, \boldsymbol{P}_k^{(1)}\right), \boldsymbol{P}_k^{(1)}\right) \\ &= c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0)) + \gamma Q^*\left(\boldsymbol{x}_{k+1}, \boldsymbol{\pi}^*\left(\boldsymbol{x}_{k+1}, \boldsymbol{P}_k^{(1)}\right), \boldsymbol{P}_k^{(1)}\right). \end{aligned} \tag{7}$$

Here, the optimal control policy is denoted by $\boldsymbol{\pi}^*(\cdot)$, hence $\boldsymbol{\pi}^*(\boldsymbol{x}_{k+1}, \boldsymbol{P}_k^{(1)}) = \boldsymbol{u}_{k+1}^*$. This Q-function is useful for solving Problem 1 as can be seen from the following lemma, which extends the relations from classical Q-learning[28] to the PRADP tracking case.

Lemma 1. *The control $\boldsymbol{u}_k$ minimizing $Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)$ is a solution for $\boldsymbol{u}_k^*$ minimizing J_k in (3) according to Problem* 1.

Proof. With (7)

$$\begin{aligned}\min_{\boldsymbol{u}_k} Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k) &= c\left(\boldsymbol{x}_k, \boldsymbol{u}_k^*, \boldsymbol{r}(\boldsymbol{P}_k, 0)\right) + \gamma Q^*\left(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}^*, \boldsymbol{P}_k^{(1)}\right)\\ &= \min_{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \dots} \sum_{i=0}^{\infty} \gamma^i c(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{r}(\boldsymbol{P}_k, i))\\ &= J_k^* \end{aligned} \tag{8}$$

■

follows, which completes the proof.

As a consequence of Lemma 1, if the Q-function $Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)$ is known, the desired optimal control $\boldsymbol{u}_k$ is given by

$$\boldsymbol{u}_k^* = \arg\min_{\boldsymbol{u}_k} Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k). \tag{9}$$

Lemma 1 and (9) reveal the meaningfulness of $Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)$ for solving Problem 1. We express this Q-function by means of linear function approximation in the following. Based on the temporal-difference (TD) error, the unknown Q-function weights can then be estimated.

3.2 | Function approximation of the tracking Q-function

As classical tabular Q-learning is unable to cope with large (or even continuous) state and control spaces, it is common to represent the Q-function, which is assumed to be smooth, by means of a linear function approximator.[29] This leads to

$$Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k) = \boldsymbol{w}^\intercal \boldsymbol{\phi}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k) + \epsilon(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k). \tag{10}$$

Here, $\boldsymbol{w} \in \mathbb{R}^q$ is the unknown optimal weight vector, $\boldsymbol{\phi} \in \mathbb{R}^q$ is a vector of activation functions, and ϵ is the approximation error. According to the Weierstrass higher-order approximation theorem,[30] a single hidden layer and appropriately smooth hidden layer activation functions $\boldsymbol{\phi}(\cdot)$ are capable of an arbitrarily accurate approximation of the Q-function. Furthermore, if $q \to \infty$, $\epsilon \to 0$.

As $\boldsymbol{w}$ is unknown a priori, let the estimated optimal Q-function be given by

$$\hat{Q}^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k) = \hat{\boldsymbol{w}}^\intercal \boldsymbol{\phi}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k). \tag{11}$$

Analogous to (9), the estimated optimal control law is defined as

$$\hat{\boldsymbol{\pi}}^*(\boldsymbol{x}_k, \boldsymbol{P}_k) = \arg\min_{\boldsymbol{u}_k} \hat{Q}^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k). \tag{12}$$

Based on this parametrization of our new Q-function, the associated TD error[26] is defined as follows.

Definition 3 (TD error of the tracking Q-function). The TD error that results from using the estimated Q-function $\hat{Q}^*(\cdot)$ (11) in the Bellman-like equation 7 is defined as

$$\begin{aligned}\delta_k &= c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0)) + \gamma \hat{Q}^*\left(\boldsymbol{x}_{k+1}, \hat{\boldsymbol{\pi}}^*\left(\boldsymbol{x}_{k+1}, \boldsymbol{P}_k^{(1)}\right), \boldsymbol{P}_k^{(1)}\right) - \hat{Q}^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)\\ &= c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0)) + \gamma \hat{\boldsymbol{w}}^\intercal \boldsymbol{\phi}\left(\boldsymbol{x}_{k+1}, \hat{\boldsymbol{\pi}}^*\left(\boldsymbol{x}_{k+1}, \boldsymbol{P}_k^{(1)}\right), \boldsymbol{P}_k^{(1)}\right) - \hat{\boldsymbol{w}}^\intercal \boldsymbol{\phi}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k). \end{aligned} \tag{13}$$

Our goal is to estimate $\hat{\boldsymbol{w}}$ in order to minimize the squared TD error δ_k^2, as the TD error quantifies the quality of the Q-function approximation. However, (13) is scalar while the unknown weight $\hat{\boldsymbol{w}} \in \mathbb{R}^q$ needs to be estimated. Thus, we utilize $N \geq q$ tuples

$$\mathcal{T}_k = \left\{c_k, \hat{Q}_k^*, \hat{Q}_k^{*+}\right\}, k = 1, \dots, N,$$

where

$$c_k = c\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0)\right),$$
$$\hat{Q}_k^* = \hat{\boldsymbol{w}}^\intercal \boldsymbol{\phi}_k = \hat{\boldsymbol{w}}^\intercal \boldsymbol{\phi}\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k\right)$$

and

$$\hat{Q}_k^{*+} = \hat{\boldsymbol{w}}^\intercal \boldsymbol{\phi}_k^+ = \hat{\boldsymbol{w}}^\intercal \boldsymbol{\phi}\left(\boldsymbol{x}_{k+1}, \hat{\boldsymbol{\pi}}^*\left(\boldsymbol{x}_{k+1}, \boldsymbol{P}_k^{(1)}\right), \boldsymbol{P}_k^{(1)}\right). \tag{14}$$

These N tuples are generated from interactions with the system in order to estimate $\hat{\boldsymbol{w}}$ using LSPI (cf. Lagoudakis and Parr[27]). Stacking (13) for the tuples $\mathcal{T}_k$, $k = 1, \dots, N$, yields

$$\underbrace{\begin{bmatrix} \delta_1 \\ \vdots \\ \delta_N \end{bmatrix}}_{\boldsymbol{\delta}} = \underbrace{\begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix}}_{\boldsymbol{c}} + \underbrace{\left(\gamma \begin{bmatrix} \boldsymbol{\phi}_1^{+\intercal} \\ \vdots \\ \boldsymbol{\phi}_N^{+\intercal} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\phi}_1^\intercal \\ \vdots \\ \boldsymbol{\phi}_N^\intercal \end{bmatrix}\right)}_{\boldsymbol{\Phi}} \hat{\boldsymbol{w}}. \tag{15}$$

If the excitation condition

$$\operatorname{rank} \boldsymbol{\Phi}^\intercal \boldsymbol{\Phi} = q \tag{16}$$

holds, a $\hat{\boldsymbol{w}}$ minimizing $\boldsymbol{\delta}^\intercal \boldsymbol{\delta}$ exists, is unique, and given by

$$\hat{\boldsymbol{w}} = \left(\boldsymbol{\Phi}^\intercal \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\intercal \boldsymbol{c}, \tag{17}$$

according to Åström and Wittenmark, theorem 2.1.[31]

Note 1. Using $\boldsymbol{P}_k^{(1)} = \boldsymbol{P}_k \boldsymbol{T}(1)$ (see (5)) in the training tuple $\mathcal{T}_k$ (14) rather than an arbitrary subsequent $\boldsymbol{P}_{k+1}$ is important as this procedure guarantees (in combination with (1)) that the Markov property holds, which is commonly required in ADP.[8]

Remark 1. The procedure described above is an extension to Lagoudakis and Parr, section 5.1,[27] to the tracking case where the minimization of the squared TD error is targeted. In addition, an alternative projection method has been implemented (cf. Lagoudakis and Parr, section 5.2[27]), which targets the approximate Q-function to be a fixed point under the Bellman operator. Both procedures yielded indistinguishable results for our LQ simulation examples, but this might be different for the general case.

Note that $\hat{\boldsymbol{\pi}}^*(\cdot)$ in $\hat{Q}_k^{*+}$ depends on $\hat{\boldsymbol{w}}$, that is, the estimation of $\hat{Q}_k^{*+}$ relies on another estimation (of the optimal control law). This mechanism is known as *bootstrapping* (cf. Sutton and Barto[24]) in reinforcement learning. As a consequence, rather than estimating $\hat{\boldsymbol{w}}$ once by means of the least-squares estimate (17), a *policy iteration* is performed starting with $\hat{\boldsymbol{w}}^{(0)}$. This procedure is given in Algorithm 1, where $e_{\hat{\boldsymbol{w}}}$ is a threshold for the terminal condition.

Note 2. Due to the use of a Q-function, which explicitly depends on the control $\boldsymbol{u}_k$, this method performs off-policy learning.[24] Thus, during training, the behavior policy (ie, the $\boldsymbol{u}_k$ that is actually applied to the system) might need to include exploration noise in order to satisfy the rank condition (16). However, due to the greedy target policy $\hat{\boldsymbol{\pi}}^*$ (cf. the policy improvement step (12)), the Q-function associated with the optimal control law is learned.

With $\hat{Q}^{(j)}(\cdot) = \hat{\boldsymbol{w}}^{(j)\intercal} \boldsymbol{\phi}(\cdot)$ and $Q^{\hat{\boldsymbol{\pi}}^{(j)}}$ according to (6), where $\boldsymbol{\pi} = \hat{\boldsymbol{\pi}}^{(j)}$, the following theorem also holds for our tracking Q-function.

Theorem 1 (Convergence of the Q-function, cf. Lagoudakis and Parr, theorem 7.1[27]). *Let $\bar{\epsilon} \geq 0$ bound the errors between the approximate Q-function $\hat{Q}^{(j)}$ and true Q-function $Q^{\hat{\boldsymbol{\pi}}^{(j)}}$ associated with $\hat{\boldsymbol{\pi}}^{(j)}$ over all iterations, that is,*

$$\left\| \hat{Q}^{(j)} - Q^{\hat{\boldsymbol{\pi}}^{(j)}} \right\|_\infty \leq \bar{\epsilon}, \forall j = 1, 2, \dots. \tag{18}$$

Then, Algorithm 1 *yields control laws such that*

$$\limsup_{j\to\infty}\left\|\hat{Q}^{(j)}-Q^{*}\right\|_{\infty}\leq\frac{2\gamma\bar{\epsilon}}{(1-\gamma)^{2}}. \tag{19}$$

Proof. The proof is adapted from Bertsekas and Tsitsiklis, proposition 6.2.[32] ■

Lagoudakis and Parr[27] point out that the appropriate choice of basis functions and the sample distribution (ie, excitation) determine the error bound $\bar{\epsilon}$. According to Theorem 1, Algorithm 1 converges to a neighborhood of the optimal tracking Q-function under an appropriate choice of basis functions $\boldsymbol{\phi}(\cdot)$ and excitation. However, for general nonlinear systems (1) and cost functions (3), an appropriate choice of basis functions and the number of neurons is "more of an art than science"[33] and remains an open problem. Furthermore, the appropriate excitation of a general nonlinear system requires that the training data covers all relevant areas of the state space in order to understand the nonlinearities of the system and to learn the weights $\boldsymbol{w}$ correctly. As the focus of this article lies on the proposal of the new Q-function for tracking purposes rather than the problem-specific tuning of neural networks and the excitation of nonlinear systems, we focus on linear systems and quadratic cost functions in the following—a setting that plays an important role in control engineering. This allows analytic insights into the structure of $Q^{*}\left(\boldsymbol{x}_k,\boldsymbol{u}_k,\boldsymbol{P}_k\right)$, and thus the proper choice of $\boldsymbol{\phi}(\cdot)$ for function approximation, in order to demonstrate the effectiveness of the proposed PRADP method.

Algorithm 1. PRADP based on LSPI

```
1: initialize $j=0, \hat{\boldsymbol{w}}^{(0)}$
2: do
3:     policy evaluation: calculate $\hat{\boldsymbol{w}}^{(j+1)}$ according to (17), where $\hat{\boldsymbol{w}}=\hat{\boldsymbol{w}}^{(j+1)}$
4:     policy improvement: obtain $\hat{\boldsymbol{\pi}}^{(j+1)}$ from (12)
5:     $j=j+1$
6: while $\left\|\hat{\boldsymbol{w}}^{(j)}-\hat{\boldsymbol{w}}^{(j-1)}\right\|_2 > e_{\hat{\mathbf{w}}}$
```

3.3 | The LQ-tracking case

In the following, assume the system dynamics

$$\boldsymbol{x}_{k+1}=\boldsymbol{A}\boldsymbol{x}_k+\boldsymbol{B}\boldsymbol{u}_k, \tag{20}$$

and the cost functional

$$\begin{aligned}J_k&=\sum_{i=0}^{\infty}\gamma^{i}\left[(\boldsymbol{x}_{k+i}-\boldsymbol{r}(\boldsymbol{P}_k,i))^{\mathsf{T}}\boldsymbol{Q}\left(\boldsymbol{x}_{k+i}-\boldsymbol{r}(\boldsymbol{P}_k,i)\right)\ +\boldsymbol{u}_{k+i}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u}_{k+i}\right]\\&=:\sum_{i=0}^{\infty}\gamma^{i}\left[\boldsymbol{e}_{k,i}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{e}_{k,i}+\boldsymbol{u}_{k+i}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u}_{k+i}\right].\end{aligned} \tag{21}$$

Here, $\boldsymbol{Q}\in\mathbb{R}^{n\times n}$ penalizes the deviation of the state $\boldsymbol{x}_{k+i}$ from the reference $\boldsymbol{r}(\boldsymbol{P}_k,i)$ and $\boldsymbol{R}\in\mathbb{R}^{m\times m}$ penalizes the control effort. Furthermore, let the following assumptions hold.

Assumption 1. Let $\boldsymbol{Q}=\boldsymbol{Q}^{\mathsf{T}}\geq\boldsymbol{0},\boldsymbol{R}=\boldsymbol{R}^{\mathsf{T}}\succ\boldsymbol{0}$, $(\boldsymbol{A},\boldsymbol{B})$ be controllable and $(\boldsymbol{C},\boldsymbol{A})$ be detectable, where $\boldsymbol{C}$ is defined such that $\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}=\boldsymbol{Q}$.

Assumption 2. Let the matrix $\boldsymbol{T}(i)$, which defines the shifted parameter matrix $\boldsymbol{P}_k^{(i)}$ according to (5) in Definition 1, be such that $|\lambda_j|<1,\forall j=1,\ldots,p$, holds, where λ_j are the eigenvalues of $\sqrt{\gamma}\boldsymbol{T}(1)$.

Note 3. Assumption 1 is typical in the LQ setting in order to ensure the existence and uniqueness of a stabilizing solution for the discrete-time algebraic Riccati equation associated with the regulation problem given by (20) and (21) for $\boldsymbol{P}_k=\boldsymbol{0}$ (cf. Kučera, theorem 8[34]), that is, if the reference trajectory is $\boldsymbol{0}$ for all k. Furthermore, it is evident that the reference trajectory $\boldsymbol{r}(\boldsymbol{P}_k,i)$ must be defined such that a controller exists, which yields finite cost J_k in order to obtain a reasonable control problem. As will be seen in Theorem 2, Assumption 2 guarantees the existence of this solution.

In order to derive the optimal control law, the tracking error $\boldsymbol{e}_{k,i}$ is first expressed as

$$\boldsymbol{e}_{k,i} = \boldsymbol{x}_{k+i} - \boldsymbol{r}(\boldsymbol{P}_k, i) = \boldsymbol{x}_{k+i} - \boldsymbol{P}_k^{(i)}\rho(0) = \underbrace{\begin{bmatrix} \boldsymbol{I}_n & \begin{bmatrix} -\rho(0) & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & -\rho(0) \end{bmatrix}^{\mathsf{T}} \end{bmatrix}}_{=:\boldsymbol{M}} \underbrace{\begin{bmatrix} \boldsymbol{x}_{k+i} \\ \boldsymbol{p}_{k,1}^{(i)} \\ \vdots \\ \boldsymbol{p}_{k,n}^{(i)} \end{bmatrix}}_{=:\boldsymbol{y}_{k,i}}, \quad i = 0, 1, \dots, \tag{22}$$

where $\boldsymbol{I}_n$ denotes the $n \times n$ identity matrix and $\boldsymbol{y}_{k,i}$ is the state $\boldsymbol{x}_{k+i}$ extended by the reference parameter $\boldsymbol{P}_k^{(i)}$. The associated optimal controller is then given by the following theorem.

Theorem 2 (Optimal tracking control law). *Let a reference (2) and a shift matrix $\boldsymbol{T}(i)$ as in Definition 1 be given. Then,*

1. *The optimal controller which minimizes (21) subject to the system dynamics (20) is linear w.r.t. the $\boldsymbol{y}_{k,i}$ in (22) and can be stated as*

$$\boldsymbol{\pi}^*(\boldsymbol{x}_{k+i}, \boldsymbol{P}_k^{(i)}) = \boldsymbol{u}_{k+i}^* = -\boldsymbol{L}\boldsymbol{y}_{k,i}, \quad i = 0, 1, \dots. \tag{23}$$

Here, the optimal gain $\boldsymbol{L}$ is given by

$$\boldsymbol{L} = (\gamma\tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{B}} + \boldsymbol{R})^{-1}\gamma\tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{A}}, \tag{24}$$

where

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}(1)^{\mathsf{T}} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{T}(1)^{\mathsf{T}} \end{bmatrix}, \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix}, \tag{25}$$

$\tilde{\boldsymbol{A}} \in \mathbb{R}^{n(p+1)\times n(p+1)}$, $\tilde{\boldsymbol{B}} \in \mathbb{R}^{n(p+1)\times m}$, $\tilde{\boldsymbol{Q}} = \boldsymbol{M}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{M}$ and where $\tilde{\boldsymbol{S}}$ denotes the solution of the discrete-time algebraic Riccati equation

$$\tilde{\boldsymbol{S}} = \gamma\tilde{\boldsymbol{A}}^{\mathsf{T}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{A}} - \gamma\tilde{\boldsymbol{A}}^{\mathsf{T}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{B}}(\boldsymbol{R} + \tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{B}})^{-1}\tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{Q}}. \tag{26}$$

2. *Furthermore, under Assumptions 1–2, the optimal controller $\boldsymbol{\pi}^*(\boldsymbol{x}_{k+i}, \boldsymbol{P}_k^{(i)})$ exists and is unique.*

Proof. 1. With (22), the discounted cost (21) can be reformulated as

$$J_k = \sum_{i=0}^{\infty} \gamma^i \left[\boldsymbol{y}_{k,i}{}^{\mathsf{T}}\boldsymbol{M}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{M}\boldsymbol{y}_{k,i} + \boldsymbol{u}_{k+i}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u}_{k+i}\right]. \tag{27}$$

Furthermore, note that with (20) and (5),

$$\boldsymbol{y}_{k,i+1} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{x}_{k+i} + \boldsymbol{B}\boldsymbol{u}_{k+i} \\ \boldsymbol{T}(1)^{\mathsf{T}}\boldsymbol{p}_{k,1}^{(i)} \\ \vdots \\ \boldsymbol{T}(1)^{\mathsf{T}}\boldsymbol{p}_{k,n}^{(i)} \end{bmatrix} = \tilde{\boldsymbol{A}}\boldsymbol{y}_{k,i} + \tilde{\boldsymbol{B}}\boldsymbol{u}_{k+i} \tag{28}$$

holds. With γ, $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{Q}}$, and $\boldsymbol{R}$, a standard *discounted* LQ-regulation problem results from (27) for the extended state $\boldsymbol{y}_{k,i}$. Considering that the discounted problem is equivalent to the undiscounted problem with $\sqrt{\gamma}\tilde{\boldsymbol{A}}$, $\sqrt{\gamma}\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{Q}}$ and $\boldsymbol{R}$ (cf. Gaitsgory et al[35]), the given problem can be reformulated to a standard undiscounted LQ problem. For the latter, it is well known that the optimal controller is linear w.r.t. the state (here the extended state $\boldsymbol{y}_{k,i}$) and the optimal gain is given by (24) (see, eg, Lewis et al, section 2.4[36]). Thus, (23) holds and the first theorem assertion follows.

2. For the second theorem assertion, we note that the stabilizability of $(\sqrt{\gamma}\tilde{\boldsymbol{A}}, \sqrt{\gamma}\tilde{\boldsymbol{B}})$ directly follows from Assumptions 1 and 2 due to $(\boldsymbol{A}, \boldsymbol{B})$ being controllable and $|\lambda_j| < 1$, $\forall j = 1, \dots, p$. In addition, $\boldsymbol{Q} \geq \boldsymbol{0}$ yields $\tilde{\boldsymbol{Q}} \geq \boldsymbol{0}$. As $(\boldsymbol{C}, \boldsymbol{A})$ is

detectable (Assumption 1), with $\tilde{\boldsymbol{C}}$ such that $\tilde{\boldsymbol{C}}^{\mathsf{T}}\tilde{\boldsymbol{C}} = \tilde{\boldsymbol{Q}}$, it follows that $(\tilde{\boldsymbol{C}}\sqrt{\gamma}\tilde{\boldsymbol{A}})$ is also detectable, because all additional states in $\tilde{\boldsymbol{A}}$ compared to $\boldsymbol{A}$ are stable due to Assumption 2. Finally, due to $\tilde{\boldsymbol{Q}} \geq \boldsymbol{0}$, $\boldsymbol{R} \succ \boldsymbol{0}$, $(\sqrt{\gamma}\tilde{\boldsymbol{A}}, \sqrt{\gamma}\tilde{\boldsymbol{B}})$ being stabilizable and $(\tilde{\boldsymbol{C}}, \sqrt{\gamma}\tilde{\boldsymbol{A}})$ being detectable, a unique stabilizing solution exists (cf. Kučera, theorem 8[34]). ■

Note 4. Theorem 2 demonstrates that in the case of *known* system dynamics by means of $\boldsymbol{A}$ and $\boldsymbol{B}$, the optimal tracking controller $\boldsymbol{L}$ can be calculated directly by solving the discrete-time algebraic Riccati equation[37] associated with $\sqrt{\gamma}\tilde{\boldsymbol{A}}$, $\sqrt{\gamma}\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{Q}}$, and $\boldsymbol{R}$.

Equation (28) also demonstrates that the important Markov property holds (cf. Note 1). As a consequence of Theorem 2, this yields the following problem in the LQ PRADP case with *unknown* system dynamics.

Problem 2. Let the system dynamics described by the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ be unknown. For $i = 0, 1, \ldots$, find the linear extended state feedback controller $\boldsymbol{L}$ (cf. (23)) minimizing the cost functional J_k (21) and apply $\boldsymbol{u}_k^* = -\boldsymbol{L}\boldsymbol{y}_{k,0}$ to the unknown system (20).

Before we derive the control law $\boldsymbol{L}$ without using the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we analyze the structure of the optimal Q-function $Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)$ associated with Problem 2 in the following lemma.

Lemma 2 (Structure of the tracking Q-function). *The Q-function associated with Problem 2 has the quadratic form*

$$Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k) = \boldsymbol{z}_k^{\mathsf{T}}\boldsymbol{H}\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \\ \boldsymbol{p}_{k,1:n} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{h}_{xx} & \boldsymbol{h}_{xu} & \boldsymbol{h}_{xp} \\ \boldsymbol{h}_{ux} & \boldsymbol{h}_{uu} & \boldsymbol{h}_{up} \\ \boldsymbol{h}_{px} & \boldsymbol{h}_{pu} & \boldsymbol{h}_{pp} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \\ \boldsymbol{p}_{k,1:n} \end{bmatrix}, \tag{29}$$

where $\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{x}_k^{\mathsf{T}} & \boldsymbol{u}_k^{\mathsf{T}} & \boldsymbol{p}_{k,1:n}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} \boldsymbol{x}_k^{\mathsf{T}} & \boldsymbol{u}_k^{\mathsf{T}} & \boldsymbol{p}_{k,1}^{\mathsf{T}} & \cdots & \boldsymbol{p}_{k,n}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$ *and* $\boldsymbol{H}$ *is chosen such that* $\boldsymbol{H} = \boldsymbol{H}^{\mathsf{T}}$.

Proof. With (6) and (7),

$$Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k) = c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0)) + \sum_{i=1}^{\infty} \gamma^i c\left(\boldsymbol{x}_{k+i}, \boldsymbol{\pi}^*\left(\boldsymbol{x}_{k+i}, \boldsymbol{P}_k^{(i)}\right), \boldsymbol{r}(\boldsymbol{P}_k, i)\right) \tag{30}$$

follows. With (20), (23), and (5), it is evident that the states $\boldsymbol{x}_{k+i}$ and controls $\boldsymbol{\pi}^*(\boldsymbol{x}_{k+i}, \boldsymbol{P}_k^{(i)})$ are linear w.r.t. $\boldsymbol{z}_k$, $\forall i = 0, 1, \ldots$. From this linear dependency and with (22), the linearity of the tracking error $\boldsymbol{e}_{k,i}$ w.r.t. $\boldsymbol{z}_k$, $\forall i = 0, 1, \ldots$ results. Due to the linear dependencies of $\boldsymbol{e}_{k,i}$ and $\boldsymbol{\pi}^*(\cdot)$ on $\boldsymbol{z}_k$ and the quadratic structure of $c(\cdot)$ in (21), the Q-function in (30) is quadratic w.r.t. $\boldsymbol{z}_k$, thus (29) holds. ■

As a consequence of Lemma 2, the optimal Q-function Q^* can be parametrized exactly by means of the function approximator $\hat{Q}^*$ using (11) if $\hat{\boldsymbol{w}} = \boldsymbol{w}$ corresponds to the nonredundant elements of $\boldsymbol{H} = \boldsymbol{H}^{\mathsf{T}}$ (elements of $\hat{\boldsymbol{w}}$ associated with off-diagonal elements of $\boldsymbol{H}$ need to be multiplied by the constant factor 2) and the activation functions are chosen as $\boldsymbol{\phi} = \boldsymbol{z}_k \otimes \boldsymbol{z}_k$. Here, $\otimes$ denotes the Kronecker product. Based on Lemma 2, the optimal control law in terms of $\boldsymbol{H}$, without using the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, is given as follows.

Theorem 3 (Optimal tracking control law in terms of $\boldsymbol{H}$). *The unique optimal extended state feedback control minimizing* J_k *(21) is given by*

$$\boldsymbol{u}_k^* = \boldsymbol{\pi}^*(\boldsymbol{x}_k, \boldsymbol{P}_k) = -\boldsymbol{L}\boldsymbol{y}_k^{\boldsymbol{P}_k} = -\boldsymbol{h}_{uu}^{-1}\begin{bmatrix} \boldsymbol{h}_{ux} & \boldsymbol{h}_{up} \end{bmatrix}\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{p}_{k,1:n} \end{bmatrix}. \tag{31}$$

Proof. According to Lemma 1, the desired control $\boldsymbol{u}_k^*$ minimizing $Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)$ also minimizes J_k. With (29) and $\boldsymbol{H} = \boldsymbol{H}^{\mathsf{T}}$, the necessary condition

$$\frac{\partial Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)}{\partial \boldsymbol{u}_k} = 2\left(\boldsymbol{h}_{ux}\boldsymbol{x}_k + \boldsymbol{h}_{up}\boldsymbol{p}_{k,1:n} + \boldsymbol{h}_{uu}\boldsymbol{u}_k\right) \overset{!}{=} \boldsymbol{0} \tag{32}$$

yields the control $\boldsymbol{u}_k^*$ given in (31). Furthermore,

$$\frac{\partial^2 Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)}{\partial \boldsymbol{u}_k^2} = 2\boldsymbol{h}_{uu} \tag{33}$$

demonstrates that $\boldsymbol{h}_{uu} \succ \boldsymbol{0}$ is required in order to ensure that the control $\boldsymbol{u}_k^*$ (31) minimizes J_k (21). This is shown by the following. If $Q^*_{\text{reg}}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is the optimal Q-function related to the regulation case, that is, where $\boldsymbol{r}(\boldsymbol{P}_k, i) = \boldsymbol{r}(\boldsymbol{0}, i) = \boldsymbol{0}$, then it is evident that

$$Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{0}) = Q^*_{\text{reg}}(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad \forall \boldsymbol{x}_k \in \mathbb{R}^n, \boldsymbol{u}_k \in \mathbb{R}^m, \tag{34}$$

must be true. Furthermore, for the regulation case, it is well known that

$$Q^*_{\text{reg}}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix}^{\intercal} \begin{bmatrix} \boldsymbol{h}_{\text{reg},xx} & \boldsymbol{h}_{\text{reg},xu} \\ \boldsymbol{h}_{\text{reg},ux} & \boldsymbol{h}_{\text{reg},uu} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix}^{\intercal} \begin{bmatrix} \gamma \boldsymbol{A}^{\intercal}\boldsymbol{S}\boldsymbol{A} + \boldsymbol{Q} & \gamma \boldsymbol{A}^{\intercal}\boldsymbol{S}\boldsymbol{B} \\ \gamma \boldsymbol{B}^{\intercal}\boldsymbol{S}\boldsymbol{A} & \gamma \boldsymbol{B}^{\intercal}\boldsymbol{S}\boldsymbol{B} + \boldsymbol{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix} \tag{35}$$

holds (see, eg, Bradtke et al[38]). Here, $\boldsymbol{S}$ is the solution of the discrete-time algebraic Riccati equation

$$\boldsymbol{S} = \gamma \boldsymbol{A}^{\intercal}\boldsymbol{S}\boldsymbol{A} - \gamma \boldsymbol{A}^{\intercal}\boldsymbol{S}\boldsymbol{B}(\boldsymbol{R} + \boldsymbol{B}^{\intercal}\boldsymbol{S}\boldsymbol{B})^{-1}\boldsymbol{B}^{\intercal}\boldsymbol{S}\boldsymbol{A} + \boldsymbol{Q}. \tag{36}$$

Under Assumption 1, $\boldsymbol{S} = \boldsymbol{S}^{\intercal} \succeq \boldsymbol{0}$ exists and is unique (see Kučera, theorem 8[34]). Thus, from (34) and (35),

$$\boldsymbol{h}_{uu} = \boldsymbol{h}_{\text{reg},uu} = \gamma \boldsymbol{B}^{\intercal}\boldsymbol{S}\boldsymbol{B} + \boldsymbol{R} \succ \boldsymbol{0} \tag{37}$$

results. This completes the proof. ∎

According to Theorem 3, if $\boldsymbol{H}$ (or equivalently $\boldsymbol{w}$) is known, both Q^* and $\boldsymbol{\pi}^*$ can be calculated. In the following, we describe how to determine $\boldsymbol{w}$ from the data tuples $\mathcal{T}_k$, $k = 1, \dots, N$ given in (14). Lemma 2 shows that $Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{P}_k)$ is quadratic w.r.t. $\boldsymbol{z}_k$ for Problem 2. Furthermore, (31) in Theorem 3 establishes the relationship between $\boldsymbol{H}$ and the optimal controller $\boldsymbol{\pi}^*(\boldsymbol{x}_k, \boldsymbol{P}_k)$. Consequently, $\boldsymbol{w}$ and thus the optimal controller $\boldsymbol{\pi}^*(\boldsymbol{x}_k, \boldsymbol{P}_k)$ can be learned from data without using the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ by employing the LSPI-based PRADP method as given in Algorithm 1. The policy improvement step is directly given by

$$\hat{\boldsymbol{\pi}}^{(j+1)}(\boldsymbol{x}_k, \boldsymbol{P}_k) = -\left(\boldsymbol{h}_{uu}^{(j+1)}\right)^{-1} \begin{bmatrix} \boldsymbol{h}_{ux}^{(j+1)} & \boldsymbol{h}_{up}^{(j+1)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{p}_{k,1:n} \end{bmatrix}. \tag{38}$$

In the next section, simulation results for this PRADP algorithm are presented.

4 | RESULTS

In order to validate our proposed PRADP tracking method, we show simulation results where the reference trajectory is parametrized by means of cubic polynomials.* Furthermore, we compare the results with an ADP tracking method that assumes that the reference can be described by a time-invariant exo-system $\boldsymbol{f}_{\text{ref}}(\boldsymbol{r}_k)$. Finally, we compare our learned controller that does not know the system dynamics with the ground truth controller, which is calculated using full system knowledge.

4.1 | Cubic polynomial reference parametrization

We choose $\boldsymbol{r}(\boldsymbol{P}_k, i)$ to be a cubic polynomial w.r.t. i, that is, $\boldsymbol{\rho}(i) = \begin{bmatrix}(iT)^3 & (iT)^2 & iT & 1\end{bmatrix}^{\intercal}$, where T denotes the sampling time. An associated transformation that is needed to obtain the shifted version $\boldsymbol{P}_k^{(i)}$ of $\boldsymbol{P}_k$ according to Definition 1 is then given by

$$\boldsymbol{r}(\boldsymbol{P}_k, i+j) = \boldsymbol{P}_k \boldsymbol{\rho}(i+j) = \boldsymbol{P}_k \begin{bmatrix} ((i+j)T)^3 \\ ((i+j)T)^2 \\ (i+j)T \\ 1 \end{bmatrix} = \boldsymbol{P}_k \underbrace{\begin{bmatrix} 1 & 3iT & 3(iT)^2 & (iT)^3 \\ 0 & 1 & 2iT & (iT)^2 \\ 0 & 0 & 1 & iT \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{T}(i)} \boldsymbol{\rho}(j) = \boldsymbol{P}_k^{(i)} \boldsymbol{\rho}(j). \tag{39}$$

*Other approximations can be used by choosing different basis functions $\boldsymbol{\rho}(i)$ (eg, linear interpolation with $\boldsymbol{\rho}(i) = \begin{bmatrix} iT & 1 \end{bmatrix}^{\intercal}$ or zero-order hold with $\boldsymbol{\rho}(i) = 1$).

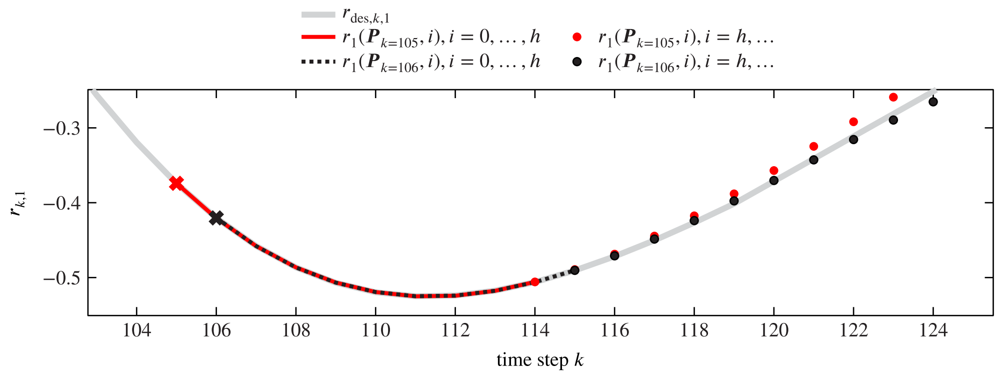


FIGURE 1 Example plot of the desired reference trajectory $r_{\text{des},k,1}$ (gray) of the first state x_1 and the cubic polynomial approximations at the time steps $k = 105$ (red) and $k = 106$ (black) resulting from $\boldsymbol{P}_{k=105}$ and $\boldsymbol{P}_{k=106}$. The horizon for fitting the cubic polynomials is $h = 10$ in this example (red solid and black dashed lines). Thus, at each time step k, the parameter $\boldsymbol{P}_k$ is fitted w.r.t. the horizon of length h [Color figure can be viewed at wileyonlinelibrary.com]

For any given desired reference trajectory $\boldsymbol{r}_{\text{des},k}$, the parameter $\boldsymbol{P}_k$ is required at each time step k such that $\boldsymbol{r}(\boldsymbol{P}_k, i)$, $i = 0, 1, \ldots$ is an approximation of $\boldsymbol{r}_{\text{des},k+i}$. In order to determine $\boldsymbol{P}_k$, we proceed as follows. Let a horizon h be given on which the desired trajectory is known during runtime, that is, $\boldsymbol{r}_{\text{des},k:k+h-1}$ is given. Then, we determine $\boldsymbol{P}_k$ by means of weighted least-squares (LS) regression. Therefore, let

$$\boldsymbol{r}_{\text{des},k:k+h-1} = \left[\boldsymbol{r}_{\text{des},k} \; \ldots \; \boldsymbol{r}_{\text{des},k+h-1}\right], \quad \boldsymbol{\rho}_{0:h-1} = \begin{bmatrix} \boldsymbol{\rho}(0)^{\mathsf{T}} \\ \boldsymbol{\rho}(1)^{\mathsf{T}} \\ \cdots \\ \boldsymbol{\rho}(h-1)^{\mathsf{T}} \end{bmatrix} \quad \text{and} \quad \boldsymbol{W}_p = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & \gamma & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \gamma^{h-1} \end{bmatrix}, \tag{40}$$

where $\boldsymbol{W}_p$ is the weighting matrix that incorporates the discount factor γ such that early time steps are more important for the fitting process compared to later time steps (cf. the definition of the cost functional J_k in (3) and (21)). Then, the reference trajectory approximation is given by the parameter

$$\boldsymbol{P}_k = \boldsymbol{r}_{\text{des},k:k+h-1} \boldsymbol{W}_p \boldsymbol{\rho}_{0:h-1} \left(\boldsymbol{\rho}_{0:h-1}^{\mathsf{T}} \boldsymbol{W}_p \boldsymbol{\rho}_{0:h-1}\right)^{-1}. \tag{41}$$

For the fitting horizon $h = 10$, which is also used in the following, example plots of the desired reference trajectory $r_{\text{des},k,1}$ and its approximations at $k = 105$ and $k = 106$ are given in Figure 1.

4.2 | Example system

In the following, an example system and a cost functional are given, which are used in order to validate our method. Consider a mass-spring-damper system

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ -\frac{c_{\text{sys}}}{m_{\text{sys}}} & -\frac{d_{\text{sys}}}{m_{\text{sys}}} \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{m_{\text{sys}}} \end{bmatrix} u(t), \tag{42}$$

with $m_{\text{sys}} = 0.5\,\text{kg}$, $c_{\text{sys}} = 0.1\ \text{Nm}^{-1}$, and $d_{\text{sys}} = 0.1\ \text{k gs}^{-1}$. Discretization of this system using Tustin approximation with $T = 0.1\,\text{s}$ yields

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 0.9990 & 0.0990 \\ -0.0198 & 0.9792 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0.0099 \\ 0.1979 \end{bmatrix} u_k. \tag{43}$$

Here, x_1 corresponds to the position and x_2 to the velocity of the mass m_{sys} while the control u_k corresponds to a force acting on the mass. The system (42), or (43), is not known to the controller and is only needed in order to generate simulation data and validate the learned controllers.

In our example, we desire to track the position of the mass (ie, x_1). Thus, we set

$$\boldsymbol{Q} = \begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{R} = 1 \tag{44}$$

to strongly penalize deviation of the first state from the parametrized reference (cf. (21)). Furthermore, we set the discount factor to $\gamma = 0.9$. In this example, the Assumptions 1 and 2 hold, that is, the optimal controller $\boldsymbol{\pi}^*(\boldsymbol{x}_{k+i}, \boldsymbol{P}_k^{(i)})$ exists and is unique according to Theorem 2.

4.3 | Simulations

In order to investigate the benefits of our proposed PRADP tracking controller, we compare our method with an ADP tracking controller from the literature,[15,16] which assumes that the reference trajectory is generated by a time-invariant exo-system $\boldsymbol{f}_{\text{ref}}(\boldsymbol{r}_k)$. Thus, following the notation of Kiumarsi et al,[16] the Q-function $Q(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}_k)$ of the exo-system-based approach depends on the current reference value $\boldsymbol{r}_k$ at time step k and not on the parameter $\boldsymbol{P}_k$ as in our PRADP method (cf. (7)). In the LQ-tracking case, their Q-function is quadratic w.r.t. $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, and $\boldsymbol{r}_k$ (cf. Kiumarsi et al,[16] section 5.1) and the Q-function-based Policy Iteration as given in Kiumarsi et al (algorithm 3)[16] can be performed.

4.3.1 | Training procedure

Both our PRADP method and the exo-system-based method from literature are trained on data tuples collected at 500 time steps. For excitation purposes, Gaussian white noise with a zero mean and a standard deviation of 1 is applied to the system input u_k for both ADP methods during data collection for training. Note that none of the methods requires the system dynamics (20) described by $\boldsymbol{A}$ and $\boldsymbol{B}$ to be known. The reference trajectory during training is given by

$$\begin{bmatrix} r_{k+1,1} \\ r_{k+1,2} \end{bmatrix} = \boldsymbol{r}_{k+1} = \boldsymbol{f}_{\text{ref}}(\boldsymbol{r}_k) = \underbrace{\begin{bmatrix} 0.9988 & 0.0500 \\ -0.0500 & 0.9988 \end{bmatrix}}_{\boldsymbol{F}_{\text{ref}}} \boldsymbol{r}_k \tag{45}$$

for the exo-system method, where the initial value is set to $\boldsymbol{r}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$, along with the associated cubic polynomials parametrized by means of $\boldsymbol{P}_k$ at each point in time for our PRADP method as described in Section 4.1. Our PRADP controller is then trained according to Algorithm 1 with the termination condition $e_{\hat{\boldsymbol{w}}} = 1 \times 10^{-6}$, using the policy improvement (38) and activation functions $\boldsymbol{\phi}(\cdot)$ that are quadratic w.r.t $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, and $\boldsymbol{p}_{k,1:n}$ as motivated by Lemma 2. We furthermore initialize $\hat{\boldsymbol{w}}^{(0)}$ such that $\hat{\boldsymbol{\pi}}^{(0)} = \boldsymbol{0}$.[†] The complete procedure is depicted in Figure 2. The comparison method is implemented as given by Kiumarsi et al (algorithm 3),[16] where we also use the terminal condition $e_{\hat{\boldsymbol{w}}} = 1 \times 10^{-6}$ in order to determine the convergence of the algorithm and set $\hat{\boldsymbol{\pi}}^{(0)} = \boldsymbol{0}$.

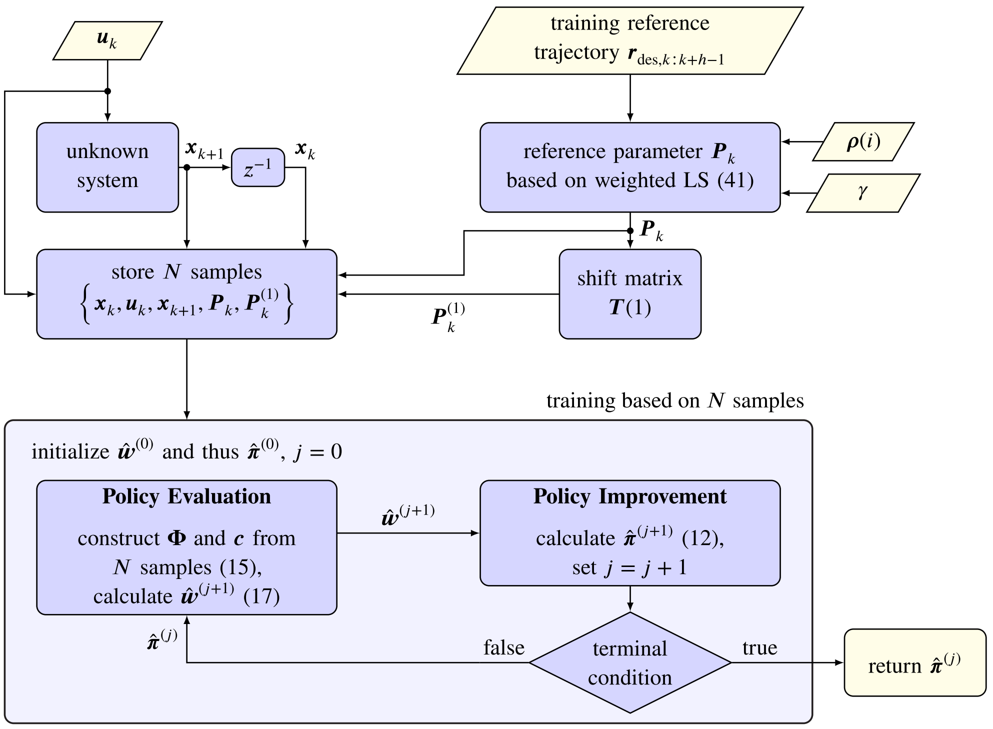
4.3.2 | Validation of the trained controllers

In order to validate the learned ADP controllers, we first compare the controllers learned from data with their ground truth solutions. Then, we give various simulation results in order to point out the differences between the PRADP controller and the controller that assumes that the reference trajectory relies on time-invariant exo-system dynamics.

Concerning the parametrized reference method, the optimal controller $\boldsymbol{L}$ calculated using the full system information (see Theorem 2 and Note 4) results in

$$\boldsymbol{\pi}^*(\boldsymbol{x}_k, \boldsymbol{P}_k) = -\underbrace{\begin{bmatrix} 6.30 & 2.26 & -0.31 & -0.97 & -2.37 & -6.40 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{L}} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{p}_{k,1:n} \end{bmatrix}. \tag{46}$$

[†]This can be done by setting the weights associated with $\boldsymbol{h}_{uu}$ such that $\boldsymbol{h}_{uu} > \boldsymbol{0}$ while all other weights are set to zero (see Lemma 2).



u_k
training reference trajectory $r_{\mathrm{des},k:k+h-1}$
unknown system
x_{k+1}
z^{-1}
x_k
reference parameter P_k based on weighted LS (41)
$\rho(i)$
γ
P_k
store N samples $\{x_k, u_k, x_{k+1}, P_k, P_k^{(1)}\}$
shift matrix $T(1)$
$P_k^{(1)}$
training based on N samples
initialize $\hat{w}^{(0)}$ and thus $\hat{\pi}^{(0)}$, $j = 0$
Policy Evaluation
construct Φ and c from N samples (15), calculate $\hat{w}^{(j+1)}$ (17)
$\hat{w}^{(j+1)}$
Policy Improvement
calculate $\hat{\pi}^{(j+1)}$ (12), set $j = j + 1$
$\hat{\pi}^{(j)}$
false
terminal condition
true
return $\hat{\pi}^{(j)}$

FIGURE 2 Flowchart describing the data collection and training procedure of the PRADP algorithm. The control input u_k during training, the reference trajectory $r_{\mathrm{des},k:k+h-1}$ during training, and the basis functions $\rho(i)$ for approximating the reference trajectory and the discount factor γ are the inputs. When the terminal condition (see Algorithm 1) is met, the optimal controller $\hat{\pi}^{(j)}$ is returned [Color figure can be viewed at wileyonlinelibrary.com]

Comparing the learned PRADP controller L_{PRADP} with this ground truth solution L yields $\|L_{\mathrm{PRADP}} - L\| = 6.51 \times 10^{-14}$. Thus, the learned controller is virtually identical to the ground truth solution which demonstrates that the optimal tracking controller has successfully been learned using the PRADP method without knowledge of the system dynamics.

For the exo-system based tracking controller, the optimal controller L_{es} (see (58) in Reference 16) results in

$$\pi^*_{\mathrm{es}}(x_k, r_k) = -\underbrace{\begin{bmatrix} 6.30 & 2.26 & -6.28 & -1.18 \end{bmatrix}}_{L_{\mathrm{es}}} \begin{bmatrix} x_k \\ r_k \end{bmatrix} \tag{47}$$

for the given example. The controller $L_{\mathrm{es,learned}}$ that is learned from data as described in Section 4.3.1 is virtually identical to the ground truth solution of the exo-system approach ($\|L_{\mathrm{es,learned}} - L_{\mathrm{es}}\| = 2.60 \times 10^{-13}$), that is, the training has been successful.

In order to compare the performance of our PRADP tracking controller with the state-of-the-art methods (which assume that the reference is generated by a time-invariant exo-system), three different scenarios are considered. In all scenarios, the controllers that have previously been trained are used without further modifications. Furthermore, the initial state is set to $x_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ in each scenario. The different scenarios are as follows:

1. The first reference trajectory is generated by the same time-invariant exo-system used during training, that is, F_{ref} as given in (45). The resulting tracking performance of the position x_1 is given in Figure 3A. The parameters $p_{k,1}$ defining the cubic polynomial associated with $r_1(P_k, 0)$ are given in Figure 3B.
2. The second reference trajectory is also generated by a time-invariant exo-system. However, exo-system dynamics other than those used for training are used. Here, the reference trajectory is generated by

$$r_{k+1} = \underbrace{\begin{bmatrix} 0.9987 & 0.0030 \\ -0.1998 & 0.9987 \end{bmatrix}}_{F_{\mathrm{ref,\,validation}}} r_k, \tag{48}$$

with $r_0 = \begin{bmatrix} 10 & 1 \end{bmatrix}^\top$. The resulting tracking performance and the parameters $p_{k,1}$ of the polynomials that approximate the reference trajectory are depicted in Figure 4. Furthermore, in order to gain insight into the tracking quality by means of the resulting cost, the immediate cost $c(x_k, u_k, r(P_k, 0))$ and the accumulated cost $\sum_{\kappa=0}^{k} c(x_\kappa, u_\kappa, r(P_\kappa, 0))$ are shown.

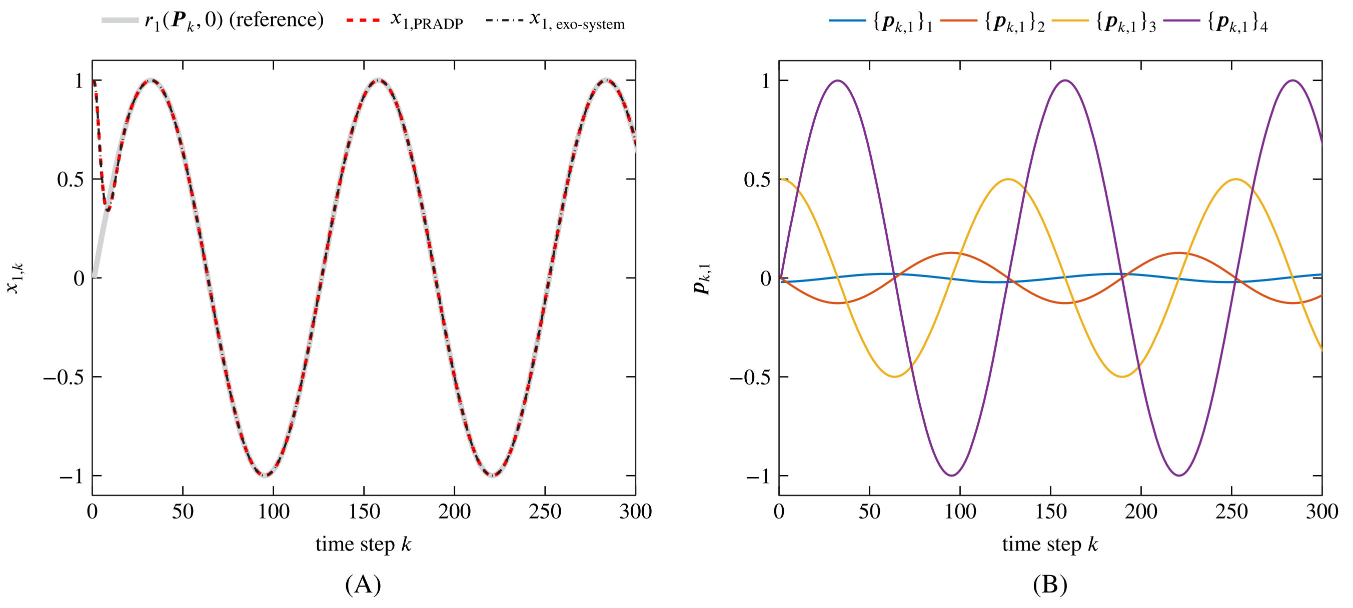


FIGURE 3 Tracking results of our proposed PRADP method compared with a state-of-the-art ADP tracking controller for Scenario 1, where the reference trajectory is generated by $\boldsymbol{F}_{\mathrm{ref}}$. A, The approximated reference trajectory $r_1(\boldsymbol{P}_k, 0)$ is depicted in gray, the tracking result of our PRADP method in red and the result of the exo-system approach in black. B, Parameter set $\boldsymbol{p}_{k,1}$ of the cubic polynomials that yield the reference trajectory $r_1(\boldsymbol{P}_k, 0)$ [Color figure can be viewed at wileyonlinelibrary.com]

3. In the third case, the reference trajectory is *not* generated by a time-invariant exo-system. Instead, some arbitrary user-defined reference trajectory is used for $r_{k,1}$, which describes the desired position of the mass. This reference trajectory is depicted in Figure 5A in gray. Furthermore, we set $r_{k,2} = 0, \forall k$ in this example (this is allowed since the second state is not penalized due to the choice of $\boldsymbol{Q}$ in (44)). The results are given in Figure 5.

4.4 | Discussion

As can be seen from Figures 3A, 4A, and 5A, our proposed method successfully tracks the parametrized reference trajectory. Because the parameter $\boldsymbol{P}_k$ not only provides the controller with the steady-state goal but also with the desired course of the trajectory (see Figure 1), the learned controller exhibits predictive rather than simply reactive behavior. This can be seen in Figures 3A, 4A, and 5A, where the system states directly follow the reference trajectories and do not lag behind. The exo-system method proposed by, for example, Luo et al[15] and Kiumarsi et al[16] results in the same tracking performance as long as the reference trajectory corresponds to the exo-system used during the training procedure (ie, as per Scenario 1, depicted in Figure 3). However, the exo-system-based approach results in major deviations from the desired trajectory as soon as the reference does not follow $\boldsymbol{F}_{\mathrm{ref}}$ anymore (ie, as soon as (45) does not hold). This can be seen in Scenario 2 (Figure 4) and Scenario 3 (Figure 5). Although the exo-system-based approach roughly follows the desired state in Scenario 3,[‡] the controller is not aware of the course of the trajectory (i.e. it cannot correctly predict the desired trajectory). Consequently, the resulting trajectory lags behind the reference trajectory in this case.

In addition, the instantaneous and accumulated costs in Figures 4C,D and 5C,D reveal that, as soon as the reference trajectory deviates from the time-invariant exo-system description $\boldsymbol{F}_{\mathrm{ref}}$, the cost of the exo-system approach drastically exceeds the cost associated with the PRADP method due to the reduced tracking performance of the former. Consequently, while the methods are comparable in Scenario 1, our method clearly outperforms the exo-system method in Scenario 2 and Scenario 3. PRADP does not require the assumption that the reference trajectory follows time-invariant exo-system dynamics but is nevertheless able to follow this kind of reference as well as all other references that can be approximated

[‡]This is a result of the state feedback term of the controller $\boldsymbol{L}_{\mathrm{es}}$ (ie, the part of the control law that is directly related to $\boldsymbol{x}_k$), which is identical to the state feedback term of the PRADP controller. This can be seen in (46) and (47).

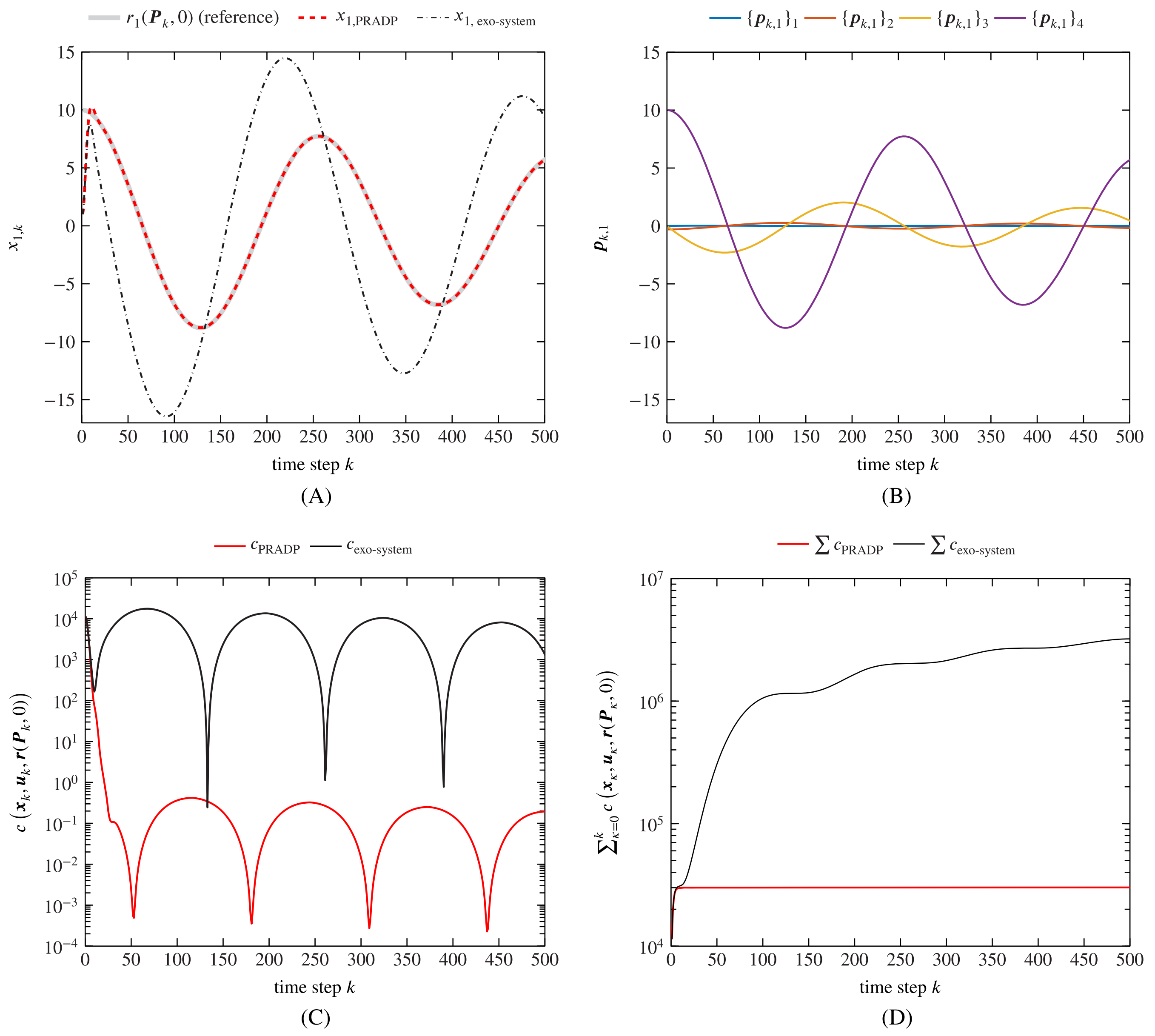


FIGURE 4 Tracking results of our proposed PRADP method compared with a state-of-the-art ADP tracking controller for Scenario 2, where the reference trajectory is generated by $\boldsymbol{F}_{\text{ref, validation}} \neq \boldsymbol{F}_{\text{ref}}$. A, The approximated reference trajectory $r_1(\boldsymbol{P}_k, 0)$ is depicted in gray, the tracking result of our PRADP method in red and the result of the exo-system approach in black. B, Parameter set $\boldsymbol{p}_{k,1}$ of the cubic polynomials that yield the reference trajectory $r_1(\boldsymbol{P}_k, 0)$. C, The instantaneous cost $c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0))$ of the PRADP method is shown in red, whereas the instantaneous cost resulting from the exo-system method is given in black. Note the logarithmic ordinate. D, The accumulated cost $\sum_{\kappa=0}^{k} c(\boldsymbol{x}_\kappa, \boldsymbol{u}_\kappa, \boldsymbol{r}(\boldsymbol{P}_\kappa, 0))$ of our method is given in red, the accumulated cost of the exo-system method is shown in black. Note the logarithmic ordinate [Color figure can be viewed at wileyonlinelibrary.com]

by means of the time-varying parameter $\boldsymbol{P}_k$. Thus, PRADP can be interpreted as a more generalized tracking approach compared to existing ADP tracking methods.

5 | CONCLUSION

In this article, we propose a new ADP-based tracking controller termed PRADP. This method implicitly incorporates the approximated reference trajectory information into the Q-function that is learned. This allows the controller to track time-varying parametrized references once the controller has been trained and does not require further adaptation or

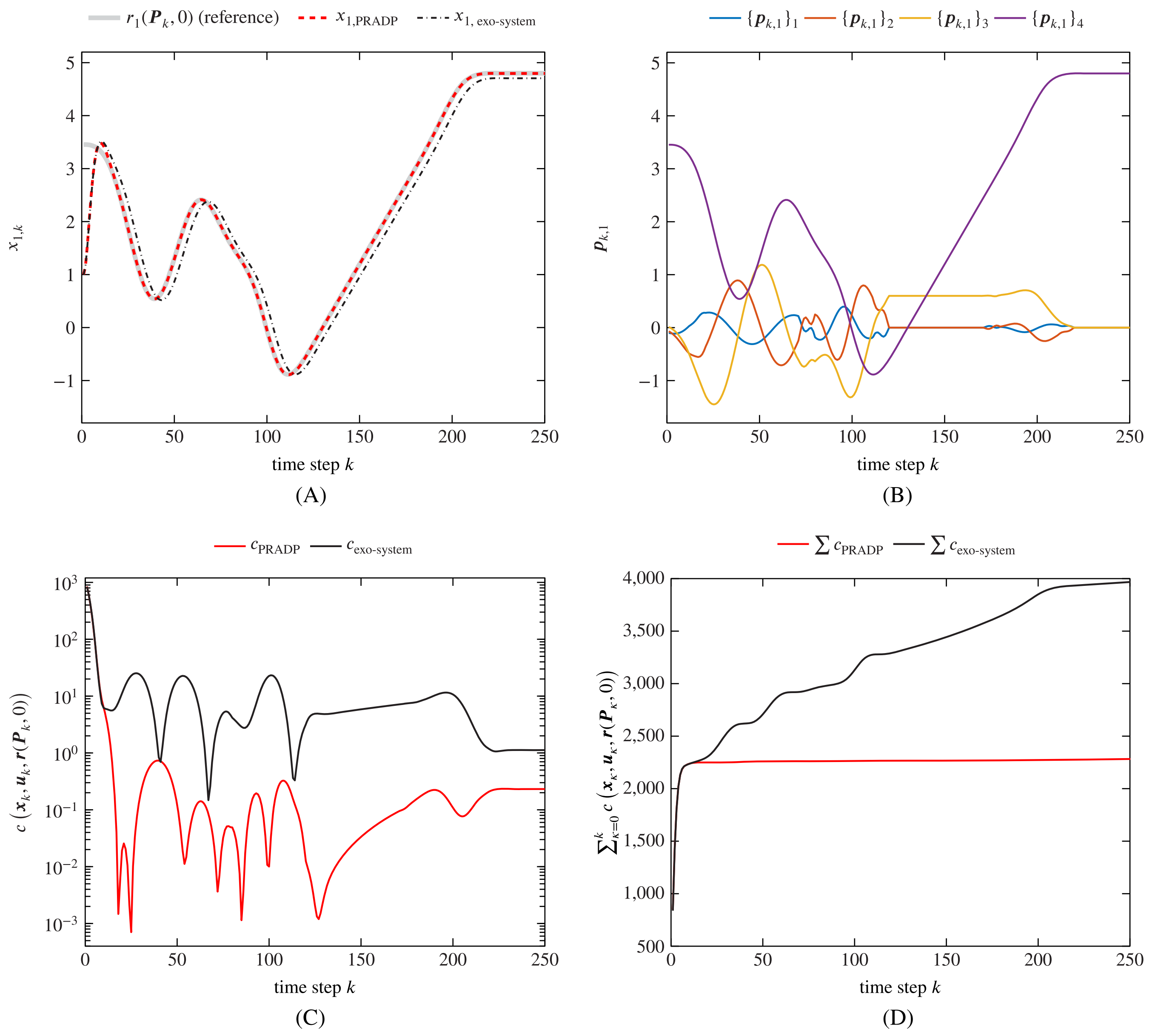


FIGURE 5 Tracking results of our proposed PRADP method compared with a state-of-the-art ADP tracking controller for Scenario 3, where the reference trajectory is not generated by a time-invariant exo-system but by some arbitrary user-defined trajectory. A, The approximated reference trajectory $r_1(\boldsymbol{P}_k, 0)$ is depicted in gray, the tracking result of our PRADP method in red and the result of the exo-system approach in black. B, Parameter set $\boldsymbol{p}_{k,1}$ of the cubic polynomials that yield the approximated reference trajectory $r_1(\boldsymbol{P}_k, 0)$. C, The instantaneous cost $c(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{r}(\boldsymbol{P}_k, 0))$ of the PRADP method is shown in red, whereas the instantaneous cost resulting from the exo-system method is given in black. Note the logarithmic ordinate. D, The accumulated cost $\sum_{\kappa=0}^{k} c(\boldsymbol{x}_\kappa, \boldsymbol{u}_\kappa, \boldsymbol{r}(\boldsymbol{P}_\kappa, 0))$ of our method is given in red, the accumulated cost of the exo-system method is shown in black [Color figure can be viewed at wileyonlinelibrary.com]

retraining compared to previous methods. Simulation results show that our learned controller is more flexible compared to state-of-the-art ADP tracking controllers which assume that the reference to be tracked follows a time-invariant exo-system. Motivated by a straightforward choice of basis functions, we concentrate on the LQ-tracking case in our simulations where the optimal controller is learned successfully. However, as the mechanism of PRADP allows more general tracking-problem formulations (see Section 3), general function approximators can be used in order to approximate Q and allow for nonlinear ADP tracking controllers in the future.

ORCID

Florian Köpf https://orcid.org/0000-0003-2536-3409

REFERENCES

1. He W, Ouyang Y, Hong J. Vibration control of a flexible robotic manipulator in the presence of input deadzone. *IEEE Trans Ind Inform*. 2017;13(1):48-59. https://doi.org/10.1109/TII.2016.2608739.
2. He W, Dong Y. Adaptive fuzzy neural network control for a constrained robot using impedance learning. *IEEE Trans Neural Netw Learn Syst*. 2018;29(4):1174-1186. https://doi.org/10.1109/TNNLS.2017.2665581.
3. Vrkalovic S, Lunca EC, Borlea ID. Model-free sliding mode and fuzzy controllers for reverse osmosis desalination plants. *Int J Artif Intell*. 2018;16(2):208-222.
4. Tan KK, Zhao S, Xu JX. Online automatic tuning of a proportional integral derivative controller based on an iterative learning control approach. *IET Control Theory Appl*. 2007;1(1):90-96. https://doi.org/10.1049/iet-cta:20050004.
5. Preitl S, Precup RE, Preitl Z, Vaivoda S, Kilyeni S, Tar JK. Iterative feedback and learning control. *Servo Syst Appl IFAC Proc Vol*. 2007;40(8):16-27. https://doi.org/10.3182/20070709-3-RO-4910.00004.
6. Casavola A, Garone E, Tedesco F. A parallel distributed supervision strategy for multi-agent networked systems. *Syst Control Lett*. 2016;97:115-124. https://doi.org/10.1016/j.sysconle.2016.09.015.
7. Lewis F, Vrabie D. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst Mag*. 2009;9(3):32-50. https://doi.org/10.1109/MCAS.2009.933854.
8. Murray JJ, Cox CJ, Lendaris GG, Saeks R. Adaptive dynamic programming. *IEEE Trans Syst Man Cybern Part C Appl Rev*. 2002;32(2):140-153. https://doi.org/10.1109/TSMCC.2002.801727.
9. Büchel M, Knoll A. Deep reinforcement learning for predictive longitudinal control of automated vehicles. Paper presented at: Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC); 2018:2391-2397.
10. Flad M, Otten J, Schwab S, Hohmann S. Steering driver assistance system: a systematic cooperative shared control design approach. Paper presented at: Proceedings of the 2014 IEEE International Conference on Systems, Man and Cybernetics (SMC); 2014:3585-3592.
11. Smets IY, Claes JE, November EJ, Bastin GP, van Impe JF. Optimal adaptive control of (bio)chemical reactors: past, present and future. *J Process Control*. 2004;14(7):795-805. https://doi.org/10.1016/j.jprocont.2003.12.005.
12. Shi W, Song S, Wu C. High-level tracking of autonomous underwater vehicles based on pseudo averaged Q-learning. Paper presented at: Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC); 2018:4138–4143.
13. Ng AY, Kim HJ, Jordan MI, Sastry S. Autonomous helicopter flight via reinforcement learning. In: Thrun S, Saul LK, Schölkopf B, eds. *Advances in Neural Information Processing Systems*. Vol 16. Cambridge, MA: MIT Press; 2004:799-806.
14. Luo B, Liu D, Huang T, Wang D. Model-free optimal tracking control via critic-only Q-learning. *IEEE Trans Neural Netw Learn Syst*. 2016;27(10):2134-2144. https://doi.org/10.1109/TNNLS.2016.2585520.
15. Kiumarsi B, Lewis FL, Modares H, Karimpour A, Naghibi-Sistani MB. Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. *Automatica*. 2014;50(4):1167-1175. https://doi.org/10.1016/j.automatica.2014.02.015.
16. Köpf F, Ebbert S, Flad M, Hohmann S. Adaptive dynamic programming for cooperative control with incomplete information. Paper presented at: Proceedings of the 2018 IEEE International Conference on Systems, Man and Cybernetics; 2018; IEEE.
17. Dierks T, Jagannathan S. Optimal tracking control of affine nonlinear discrete-time systems with unknown internal dynamics. Paper presented at: Proceedings of the 2009 Joint 48th IEEE Conference on Decision and Control (CDC) and 28th Chinese Control Conference (CCC); 2009:6750–6755; IEEE.
18. Modares H, Lewis FL. Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning. *IEEE Trans Autom Control*. 2014;59(11):3051-3056. https://doi.org/10.1109/TAC.2014.2317301.
19. Mu C, Ni Z, Sun C, He H. Data-driven tracking control with adaptive dynamic programming for a class of continuous-time nonlinear systems. *IEEE Trans Cybern*. 2017;47(6):1460-1470. https://doi.org/10.1109/TCYB.2016.2548941.
20. Zhang K, Zhang H, Xiao G, Su H. Tracking control optimization scheme of continuous-time nonlinear system via online single network adaptive critic design method. *Neurocomputing*. 2017;251:127-135. https://doi.org/10.1016/j.neucom.2017.04.008.
21. van Nieuwstadt MJ. Trajectory Generation for Nonlinear Control Systems (Dissertation). Cambridge: CA: California Institute of Technology, 1997.
22. Kiumarsi B, Lewis FL, Levine DS. Optimal control of nonlinear discrete time-varying systems using a new neural network approximation structure. *Neurocomputing*. 2015;156:157-165. https://doi.org/10.1016/j.neucom.2014.12.067.
23. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA: MIT Press; 2018.
24. Köpf F, Westermann J, Flad M, Hohmann S. Adaptive optimal control for reference tracking independent of exo-system dynamics; 2019. arXiv e-prints: arXiv:1906.05085.
25. Köpf F, Ramsteiner S, Flad M, Hohmann S. Adaptive dynamic programming for model-free tracking of trajectories with time-varying parameters; 2019. arXiv preprint: arXiv:1909.07239.
26. Sutton RS. Learning to predict by the methods of temporal differences. *Mach Learn*. 1988;3(1):9-44. https://doi.org/10.1023/A:1022633531479.
27. Lagoudakis MG, Parr R. Least-squares policy iteration. *J Mach Learn Res*. 2003;4:1107-1149.
28. Watkins CJ, Dayan P. Q-learning. *Mach Learn*. 1992;8(3-4):279-292.
29. Buşoniu L, Babuška R, Schutter DB, Ernst D. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Vol 39. Boca Raton, FL: CRC press; 2010.
30. Hornik K, Stinchcombe M, White H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw*. 1990;3(5):551-560. https://doi.org/10.1016/0893-6080(90)90005-6.
31. Åström KJ, Wittenmark B. *Adaptive Control*. 2nd ed. Boston, MA: Addison-Wesley Reading; 1995.

32. Bertsekas DP, Tsitsiklis JM. *Neuro-Dynamic Programming*. Massachusetts Athena Scientific: Belmont, CA; 1996.
33. Wang D, He H, Liu D. Adaptive critic nonlinear robust control: a survey. *IEEE Trans Cybern*. 2017;47(10):3429-3451. https://doi.org/10.1109/TCYB.2017.2712188.
34. Kučera V. The discrete Riccati equation of optimal control. *Kybernetika*. 1972;8(5):430-447.
35. Gaitsgory V, Grüne L, Höger M, Kellett CM, Weller SR. Stabilization of strictly dissipative discrete time systems with discounted optimal control. *Automatica*. 2018;93:311-320. https://doi.org/10.1016/j.automatica.2018.03.076.
36. Lewis FL, Vrabie DL, Syrmos VL. *Optimal Control*. 3rd ed. Hoboken, NJ: Wiley; 2012.
37. Arnold WF, Laub AJ. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proc IEEE*. 1984;72(12):1746-1754. https://doi.org/10.1109/PROC.1984.13083.
38. Bradtke SJ, Ydstie BE, Barto AG. Adaptive linear quadratic control using policy iteration. Paper presented at: Proceedings of the 1994 American Control Conference; 1994:3475-3479.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Köpf F, Ramsteiner S, Puccetti L, Flad M, Hohmann S. Adaptive dynamic programming for model-free tracking of trajectories with time-varying parameters. *Int J Adapt Control Signal Process*. 2020;1–18. https://doi.org/10.1002/acs.3106