Reinforcement Learning for Partially Observable Dynamic Processes: Adaptive Dynamic Programming Using Measured Output Data

F. L. Lewis, *Fellow, IEEE*, and Kyriakos G. Vamvoudakis, *Member, IEEE*

***Abstract*—Approximate dynamic programming (ADP) is a class of reinforcement learning methods that have shown their importance in a variety of applications, including feedback control of dynamical systems. ADP generally requires full information about the system internal states, which is usually not available in practical situations. In this paper, we show how to implement ADP methods using only measured input/output data from the system. Linear dynamical systems with deterministic behavior are considered herein, which are systems of great interest in the control system community. In control system theory, these types of methods are referred to as output feedback (OPFB). The stochastic equivalent of the systems dealt with in this paper is a class of partially observable Markov decision processes. We develop both policy iteration and value iteration algorithms that converge to an optimal controller that requires only OPFB. It is shown that, similar to *Q*-learning, the new methods have the important advantage that knowledge of the system dynamics is not needed for the implementation of these learning algorithms or for the OPFB control. Only the order of the system, as well as an upper bound on its "observability index," must be known. The learned OPFB controller is in the form of a polynomial autoregressive moving-average controller that has equivalent performance with the optimal state variable feedback gain.**

***Index Terms*—Approximate dynamic programming (ADP), data-based optimal control, policy iteration (PI), output feedback (OPFB), value iteration (VI).**

I. Introduction

REINFORCEMENT learning (RL) is a class of methods used in machine learning to methodically modify the actions of an agent based on observed responses from its environment [7], [33]. RL methods have been developed starting from learning mechanisms observed in mammals [4], [7]–[9], [24], [28], [29], [33]. Every decision-making organism interacts with its environment and uses those interactions to improve its own actions in order to maximize the positive effect of its limited available resources; this, in turn, leads to better survival chances. RL is a means of *learning optimal behaviors by observing the response from the environment to nonoptimal control policies*. In engineering terms, RL refers to the learning approach of an actor or agent that modifies its actions, or control policies, based on stimuli received in response to its interaction with its environment. This is based on evaluative information from the environment and could be called *action-based learning*. RL can be applied where standard supervised learning is not applicable, and requires less *a priori* knowledge. In view of the advantages offered by RL methods, a recent objective of control system researchers is to introduce and develop RL techniques that result in optimal feedback controllers for dynamical systems that can be described in terms of ordinary differential equations [4], [30], [38], [41], [42]. This includes most of the human-engineered systems, including aerospace systems, vehicles, robotic systems, and many classes of industrial processes.

Optimal control is generally an offline design technique that requires full knowledge of the system dynamics, e.g., in the linear system case, one must solve the Riccati equation. On the other hand, adaptive control is a body of online design techniques that use measured data along system trajectories to learn to compensate for unknown system dynamics, disturbances, and modeling errors to provide guaranteed performance. Optimal adaptive controllers have been designed using indirect techniques, whereby the unknown plant is first identified and then a Riccati equation is solved [12]. Inverse adaptive controllers have been provided that optimize a performance index, meaningful but not of the designer's choice [13], [20]. Direct adaptive controllers that converge to optimal solutions for unknown systems given a PI selected by the designer have generally not been developed.

Applications of RL to feedback control are discussed in [25], [28], [43], and in the recent IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS (SMC-B) Special Issue [16]. Recent surveys are given by Lewis and Vrabie [19] and Wang et al. [38]. Temporal difference RL methods [4], [7], [24], [28], [33], [41], namely, policy iteration (PI) and value iteration (VI), have been developed to solve online the Hamilton–Jacobi–Bellman (HJB) equation associated with the optimal control problem. Such methods require measurement of the entire state vector of the dynamical system to be controlled.

PI refers to a class of algorithms built as a two-step iteration: *policy evaluation* and *policy improvement*. Instead of trying a direct approach to solving the HJB equation, the PI algorithm starts by evaluating the cost/value of a given initial admissible (stabilizing) controller. The cost associated with this policy

Manuscript received October 19, 2009; revised February 10, 2010; accepted February 16, 2010. Date of publication March 29, 2010; date of current version January 14, 2011. This work was supported in part by National Science Foundation Grant ECCS-0801330 and in part by Army Research Office Grant W91NF-05-1-0314. This paper was recommended by Associate Editor S. E. Shimony.

The authors are with the Automation and Robotics Research Institute, The University of Texas at Arlington, Fort Worth, TX 76118 USA (e-mail: lewis@arri.uta.edu; kyriakos@arri.uta.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2010.2043839

is then used to obtain a new improved control policy (i.e., a policy that will have a lower associated cost than the previous one). This is often accomplished by minimizing a Hamiltonian function with respect to the new cost. The resulting policy is thus obtained based on a *greedy policy update* with respect to the new cost. These two steps of policy evaluation and policy improvement are repeated until the policy improvement step no longer changes the actual policy, and convergence to the optimal controller is achieved. One must note that the infinite horizon cost associated with a given policy can only be evaluated in the case of an admissible control policy, meaning that the control policy must be stabilizing. The difficulty of the algorithm comes from the required effort of finding an initial admissible control policy.

VI algorithms, as introduced by Werbos, are actor–critic online learning algorithms that solve the optimal control problem without the requirement of an initial stabilizing control policy [40]–[42]. Werbos referred to the family of VI algorithms as approximate dynamic programming (ADP) algorithms. These algorithms use a critic neural network (NN) for value function approximation (VFA) and an actor NN for the approximation of the control policy. Actor–critic designs using NNs for the implementation of VI are detailed in [25]. Furthermore, the SMC-B Special Issue gives a good overview on applications of ADP and RL to feedback control [16].

Most of the PI and VI algorithms require at least some knowledge of the system dynamics and measurement of the entire internal state vector that describes the dynamics of the system/environment (e.g., [2], [7], [24], [28], [33], [35]–[37], and [41]). The so-called Q-learning class of algorithms [5], [39] (called action-dependent heuristic dynamic programming (HDP) by Werbos [41], [42]) does not require exact or explicit description of the system dynamics, but they still use full state measurement in the feedback control loop. From a control system engineering perspective, this latter requirement may be hard to fulfill as measurements of the entire state vector may not be available and/or may be difficult and expensive to obtain. Although various control algorithms (e.g., state feedback) require full state knowledge, in practical implementations, taking measurements of the entire state vector is not feasible. The state vector is generally estimated based on limited information about the system available by measuring the system's outputs. State estimation techniques have been proposed (e.g., [18], [21], and [26]). These generally require a known model of the system dynamics.

In real life, it is difficult to design and implement optimal estimators because the system dynamics and the noise statistics are not exactly known. However, information about the system and noise is included in a long-enough set of input/output data. It would be desirable to be able to design an estimator by using input/output data without any system knowledge or noise identification. Such techniques belong to the field of data-based control techniques, where the control input depends on input/output data measured directly from the plant. These techniques are as follows: data-based predictive control [22], unfalsified control [27], Markov data-based linear quadratic Gaussian control [31], disturbance-based control [34], simultaneous perturbation stochastic approximation [32], pulse-response-based control [3], iterative feedback tuning [11], and virtual reference feedback tuning [15]. In [1], data-based optimal control was achieved through identification of Markov parameters.

In this paper, novel output-feedback (OPFB) ADP algorithms are derived for affine in the control input linear time-invariant (LTI) deterministic systems. Such systems have, as stochastic equivalent, the partially observable Markov decision processes (POMDPs). In this paper, data-based optimal control is implemented online using novel PI and VI ADP algorithms that require only reduced measured information available at the system outputs. These two classes of OPFB algorithms *do not require any knowledge of the system dynamics* (A, B, C) and, as such, are similar to Q-learning [5], [39], [41], [42], but they have an added advantage of requiring only measurements of input/output data and not the full system state. In order to ensure that the data set is sufficiently rich and linearly independent, there is a need to add (cf. [5]) probing noise to the control input. We discuss this issue showing that probing noise leads to bias. Adding a discount factor in the cost minimizes it to an almost-zero effect. This discount factor is related to adding exponential data weighting in the Kalman filter to remove the bias effects of unmodeled dynamics [18].

This paper is organized as follows. Section II provides the background of optimal control problem, dynamic programming, and RL methods for the linear quadratic regulation problem. Section III introduces the new class of PI and VI algorithms by formulating the temporal difference error with respect to the observed data and redefining the control sequence as the output of a dynamic polynomial controller. Section IV discusses the implementation aspects of the OPFB ADP algorithms. For convergence, the new algorithms require persistently exciting probing noise whose bias effect is canceled by using a discounted cost function. Section V presents the simulation results obtained using the new data-based ADP algorithms and is followed by concluding remarks.

II. Background

In this section, we give a review of optimal control, dynamic programming, and RL methods [i.e., PI and VI] for the linear quadratic regulator (LQR). It is pointed out that both of these methods employ contraction maps to solve the Bellman equation, which is a fixed-point equation [23]. Both PI and VI methods require *full measurements of the entire state vector*. In the next section, we will show how to implement PI and VI using only reduced information available at the system outputs.

A. Dynamic Programming and the LQR

Consider the linear TI discrete-time (DT) system

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

where $x_k \in R^n$ is the state, $u_k \in R^m$ is the control input, and $y_k \in R^p$ is the measured output. Assume throughout that (A, B) is controllable and (A, C) is observable [17].

Given a stabilizing control policy $u_k = \mu(x_k)$, associate to the system the performance index

$$V^{\mu}(x_k) = \sum_{i=k}^{\infty} \left(y_i^T Q y_i + u_i^T R u_i \right) \equiv \sum_{i=k}^{\infty} r_i \tag{2}$$

with weighting matrices $Q = Q^T \geq 0$, $R = R^T > 0$, and $(A, C\sqrt{Q})$ being observable. Note that $u_k = \mu(x_k)$ is the fixed policy. The utility is

$$r_k = y_k^T Q y_k + u_k^T R u_k. \tag{3}$$

The optimal control problem [17] is to find the policy $u_k = \mu(x_k)$ that minimizes the cost (2) along the trajectories of the system (1). Due to the special structure of the dynamics and the cost, this is known as the LQR problem.

A difference equation that is equivalent to (2) is given by the Bellman equation

$$V^{\mu}(x_k) = y_k^T Q y_k + u_k^T R u_k + V^{\mu}(x_{k+1}). \tag{4}$$

The optimal cost, or value, is given by

$$V^*(x_k) = \min_{\mu} \sum_{i=k}^{\infty} \left(y_i^T Q y_i + u_i^T R u_i \right). \tag{5}$$

According to Bellman's optimality principle, the value may be determined using the HJB equation

$$V^*(x_k) = \min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + V^*(x_{k+1}) \right) \tag{6}$$

and the optimal control is given by

$$\mu^*(x_k) = \arg\min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + V^*(x_{k+1}) \right). \tag{7}$$

For the LQR case, any value is quadratic in the state so that the cost associated to any policy $u_k = \mu(x_k)$ (not necessarily optimal) is

$$V^{\mu}(x_k) = x_k^T P x_k \tag{8}$$

for some $n \times n$ matrix P. Substituting this into (4), one obtains the LQR Bellman equation

$$x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1}. \tag{9}$$

If the policy is a linear state variable feedback so that

$$u_k = \mu(x_k) = -K x_k \tag{10}$$

then the closed-loop system is

$$x_{k+1} = (A - BK) x_k \equiv A_c x_k. \tag{11}$$

Inserting these equations into (9) and averaging over all state trajectories yield the Lyapunov equation

$$0 = (A - BK)^T P (A - BK) - P + C^T Q C + K^T R K. \tag{12}$$

If the feedback K is stabilizing and (A, C) is observable, there exists a positive definite solution to this equation. Then, the Lyapunov solution gives the value of using the state feedback K, i.e., the solution of this equation gives the kernel P such that $V^{\mu}(x_k) = x_k^T P x_k$.

To find the optimal control, insert (1) into (9) to obtain

$$x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + (A x_k + B u_k)^T P (A x_k + B u_k). \tag{13}$$

To determine the minimizing control, set the derivative with respect to u_k equal to zero to obtain

$$u_k = -(R + B^T P B)^{-1} B^T P A x_k \tag{14}$$

whence substitution into (12) yields the Riccati equation

$$0 = A^T P A - P + C^T Q C - A^T P B (R + B^T P B)^{-1} B^T P A. \tag{15}$$

This is the LQR that is equivalent to the HJB equation (6).

B. Temporal Difference, PI, and VI

It is well known that the optimal value and control can be determined online in real time using temporal difference RL methods [7], [24], [28], [33], [41], which rely on solving online for the value that makes small the so-called Bellman temporal difference error

$$e_k = -V^{\mu}(x_k) + y_k^T Q y_k + u_k^T R u_k + V^{\mu}(x_{k+1}). \tag{16}$$

The temporal difference error is defined based on the Bellman equation (4). For use in any practical real-time algorithm, the value should be approximated by a parametric structure [4], [41], [42].

For the LQR case, the value is quadratic in the state, and the Bellman temporal difference error is

$$e_k = -x_k^T P x_k + y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1}. \tag{17}$$

Given a control policy, the solution of this equation is equivalent to solving the Lyapunov equation (12) and gives the kernel P such that $V^{\mu}(x_k) = x_k^T P x_k$.

Write (6) equivalently as

$$0 = \min_{u_k} \left(-V^*(x_k) + y_k^T Q y_k + u_k^T R u_k + V^*(x_{k+1}) \right). \tag{18}$$

This is a fixed-point equation. As such, it can be solved by the method of successive approximation using a contraction map. The successive approximation method resulting from this fixed-point equation is known as PI, an iterative method of determining the optimal value and policy. For the LQR, PI is performed by the following two steps based on the temporal difference error (17) and a policy update step based on (7).

Algorithm 1—PI

Select a stabilizing initial control policy $u_k^0 = \mu^0(x_k)$. Then, for $j = 0, 1, \ldots$, perform until convergence:

1. **Policy Evaluation:** Using the policy $u_k^j = \mu^j(x_k)$ in (1), solve for P^{j+1} such that

$$0 = -x_k^T P^{j+1} x_k + y_k^T Q y_k + \left(u_k^j \right)^T R u_k^j + x_{k+1}^T P^{j+1} x_{k+1}. \tag{19}$$

2. **Policy Improvement:**

$$u_k^{j+1} = \mu^{j+1}(x_k) = \arg\min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1} \right). \quad (20)$$

■

The solution in the policy evaluation step is generally carried out in a least squares (LS) sense. The initial policy is required to be stable since only then does (19) have a meaningful solution.

This algorithm is equivalent to the following, which uses the Lyapunov equation (12), instead of the Bellman equation, and (14).

Algorithm 2—PI Lyapunov Iteration Equivalent

Select a stabilizing initial control policy K^0. Then, for $j = 0, 1, \ldots$, perform until convergence:

1. **Policy Evaluation:**

$$0 = (A - BK^j)^T P^{j+1} (A - BK^j) - P^{j+1} + C^T Q C + (K^j)^T R K^j. \quad (21)$$

2. **Policy Improvement:**

$$K^{j+1} = (R + B^T P^{j+1} B)^{-1} B^T P^{j+1} A. \quad (22)$$

■

It was shown in [4] that, under general conditions, the policy μ^{j+1} obtained by (20) is improved over μ^j in the sense that $V^{\mu^{j+1}}(x_k) \leq V^{\mu^j}(x_k)$. It was shown by Hewer [10] that Algorithm 2 converges under the controllability/observability assumptions if the initial feedback gain is stabilizing.

Note that, in PI Algorithm 2, the system dynamics (A, B) are required for the policy evaluation step, while in PI Algorithm 1, they are not. Algorithm 2 is performed offline knowing the state dynamics.

On the other hand, Algorithm 1 is performed online in real time as the data (x_k, r_k, x_{k+1}) are measured at each time step, with $r_k = y_k^T Q y_k + u_k^T R u_k$ being the utility. Note that (19) is a scalar equation, whereas the value kernel P is a symmetric $n \times n$ matrix with $n(n+1)/2$ independent elements. Therefore, $n(n+1)/2$ data sets are required before (19) can be solved. This is a standard problem in LS estimation. The policy evaluation step may be performed using batch LS, as enough data are collected along the system trajectory, or using recursive LS (RLS). The dynamics (A, B) are not required for this, since the state x_{k+1} is measured at each step, which contains implicit information about the dynamics (A, B). This procedure amounts to a stochastic approximation method that evaluates the performance of a given policy along one sample path, e.g., the system trajectory. PI Algorithm 1 effectively provides a method for solving the Riccati equation (15) online using data measured along the system trajectories. Full state measurements of x_k are required.

A second class of algorithms for the online iterative solution of the optimal control problem based on the Bellman temporal difference error (17) is given by VI or HDP [41]. Instead of the fixed-point equation in the form of (18), which leads to PI, consider (6)

$$V^*(x_k) = \min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + V^*(x_{k+1}) \right) \quad (23)$$

which is also a fixed-point equation. As such, it can be solved using a contraction map by successive approximation using the VI algorithm.

Algorithm 3—VI

Select an initial control policy K^0. Then, for $j = 0, 1, \ldots$, perform until convergence:

1. **Value Update:**

$$x_k^T P^{j+1} x_k = y_k^T Q y_k + \left(u_k^j \right)^T R u_k^j + x_{k+1}^T P^j x_{k+1}. \quad (24)$$

2. **Policy Improvement:**

$$u_k^{j+1} = \mu^{j+1}(x_k) = \arg\min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1} \right). \quad (25)$$

■

Since (23) is not an equation but a recursion, its implementation does not require a stabilizing policy. Therefore, the VI algorithm does not require an initial stabilizing gain. It is equivalent to a matrix recursion knowing the system dynamics (A, B), which has been shown to converge in [14].

Note that the policy update steps in PI Algorithm 1 and VI Algorithm 3 rely on the hypothesis of the VFA parameterization (8). Both algorithms require knowledge of the dynamics (A, B) for the policy improvement step. In [2], it is shown that if a second approximator structure is assumed for the policy, then only the B matrix is needed for the policy improvement in Algorithm 1 or 3.

An approach that provides online real-time algorithms for the solution of the optimal control problem without knowing any system dynamics is Q-learning. This has been applied to both PI [5] and VI, where it is known as action-dependent HDP [41].

All these methods require measurement of the full state $x_k \in R^n$.

III. Temporal Difference, PI, and VI Based on OPFB

This section presents the new results of this paper. The Bellman error (17) for LQR is quadratic in the state. This can be used in a PI or VI algorithm for online learning of optimal controls as long as full measurements of state $x_k \in R^n$ are available. In this section, we show how to write the Bellman temporal difference error in terms *only of the observed data*, namely, the input sequence u_k and the output sequence y_k.

The main results of the following equations are (45) and (46), which give a temporal difference error in terms only of the observed output data.

To reformulate PI and VI in terms only of the observed data, we show how to write $V^{\mu}(x_k) = x_k^T P x_k$ as a quadratic form in terms of the input and output sequences. A surprising benefit is that there result two algorithms for RL that *do not require any knowledge of the system dynamics* (A, B, C). That is, these algorithms have the same advantage as Q-learning in not requiring knowledge of the system dynamics, yet they have the added benefit of requiring only measurements of the available input/output data, not the full system state as required by Q-learning.

A. Writing the Value Function in Terms of Available Measured Data

Consider the deterministic linear TI system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{26}$$

with $x_k \in R^n$, $u_k \in R^m$, and $y_k \in R^p$. Assume that (A, B) is controllable and (A, C) is observable [17]. Controllability is the property of matrices (A, B) and means that any initial state can be driven to any desired final state. Observability is a property of (A, C) and means that observations of the output y_k over a long-enough time horizon can be used to reconstruct the full state x_k. Given the current time k, the dynamics can be written on a time horizon $[k-N, k]$ as the expanded state equation

$$x_k = A^N x_{k-N} + \begin{bmatrix} B & AB & A^2B & \cdots & A^{N-1}B \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} \tag{27}$$

$$\begin{bmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-N} \end{bmatrix} = \begin{bmatrix} CA^{N-1} \\ \vdots \\ CA \\ C \end{bmatrix} x_{k-N} + \begin{bmatrix} 0 & CB & CAB & \cdots & CA^{N-2}B \\ 0 & 0 & CB & \cdots & CA^{N-3}B \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & & 0 & CB \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} \tag{28}$$

or by appropriate definition of variables as

$$x_k = A^N x_{k-N} + U_N \overline{u}_{k-1,k-N} \tag{29}$$

$$\overline{y}_{k-1,k-N} = V_N x_{k-N} + T_N \overline{u}_{k-1,k-N} \tag{30}$$

with $U_N = \begin{bmatrix} B & AB & \cdots & A^{N-1}B \end{bmatrix}$ being the controllability matrix and

$$V_N = \begin{bmatrix} CA^{N-1} \\ \vdots \\ CA \\ C \end{bmatrix} \tag{31}$$

being the observability matrix, where $V_N \in R^{pN \times n}$. T_N is the Toeplitz matrix of Markov parameters.

Vectors

$$\overline{y}_{k-1,k-N} = \begin{bmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-N} \end{bmatrix} \in R^{pN}$$

$$\overline{u}_{k-1,k-N} = \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} \in R^{mN}$$

are the input and output sequences over the time interval $[k-N, k-1]$. They represent the available measured data.

Since (A, C) is observable, there exists a K, the observability index, such that $rank(V_N) < n$ for $N < K$ and that $rank(V_N) = n$ for $N \geq K$. Note that K satisfies $Kp \geq n$. Let $N \geq K$. Then, V_N has full column rank n, and there exists a matrix $M \in R^{n \times pN}$ such that

$$A^N = MV_N. \tag{32}$$

This was used in [1] to find the optimal control through identification of Markov parameters.

Since V_N has full column rank, its left inverse is given as

$$V_N^+ = \left(V_N^T V_N\right)^{-1} V_N^T \tag{33}$$

so that

$$M = A^N V_N^+ + Z\left(I - V_N V_N^+\right) \equiv M_0 + M_1 \tag{34}$$

for any matrix Z, with M_0 denoting the minimum norm operator and $P(R^{\perp}(V_N)) = I - V_N V_N^+$ being the projection onto a range perpendicular to V_N.

The following lemma shows how to write the system state in terms of the input/output data.

Lemma 1: Let the system (26) be observable. Then, the system state is given uniquely in terms of the measured input/output sequences by

$$\begin{aligned} x_k &= M_0 \overline{y}_{k-1,k-N} + (U_N - M_0 T_N)\overline{u}_{k-1,k-N} \\ &\equiv M_y \overline{y}_{k-1,k-N} + M_u \overline{u}_{k-1,k-N} \end{aligned} \tag{35}$$

or

$$x_k = \begin{bmatrix} M_u & M_y \end{bmatrix} \begin{bmatrix} \overline{u}_{k-1,k-N} \\ \overline{y}_{k-1,k-N} \end{bmatrix} \tag{36}$$

where $M_y = M_0$ and $M_u = U_N - M_0 T_N$, with $M_0 = A^N V_N^+$, $V_N^+ = (V_N^T V_N)^{-1} V_N^T$, being the left inverse of the observability matrix (31), and $N \geq K$, where K is the observability index.

Proof: Note that $A^N x_{k-N} = MV_N x_{k-N}$ so that, according to (30)

$$\begin{aligned} A^N x_{k-N} &= MV_N x_{k-N} \\ &= M\overline{y}_{k-1,k-N} - MT_N \overline{u}_{k-1,k-N} \end{aligned} \tag{37}$$

$$\begin{aligned} (M_0 + M_1)V_N x_{k-N} &= (M_0 + M_1)\overline{y}_{k-1,k-N} \\ &\quad - (M_0 + M_1)T_N \overline{u}_{k-1,k-N}. \end{aligned} \tag{38}$$

Note, however, that $M_1 V_N = 0$ so that $MV_N x_{k-N} = M_0 V_N x_{k-N}$, and apply M_1 to (30) to see that

$$0 = M_1 \overline{y}_{k-1,k-N} - M_1 T_N \overline{u}_{k-1,k-N} \qquad \forall M_1 \text{ s.t. } M_1 V_N = 0. \tag{39}$$

Therefore

$$A^N x_{k-N} = M_0 V_N x_{k-N} = M_0 \overline{y}_{k-1,k-N} - M_0 T_N \overline{u}_{k-1,k-N} \tag{40}$$

independently of M_1. Then, from (29)

$$\begin{aligned} x_k &= M_0 \overline{y}_{k-1,k-N} + (U_N - M_0 T_N) \overline{u}_{k-1,k-N} \\ &\equiv M_y \overline{y}_{k-1,k-N} + M_u \overline{u}_{k-1,k-N}. \end{aligned} \tag{41}$$

■

This result expresses x_k in terms of the system inputs and outputs from time $k-N$ to time $k-1$. Now, we will express the value function in terms of the inputs and outputs.

It is important to note that the system dynamics information (e.g., A, B, and C) must be known to use (36). In fact, $M_y = M_0$ is given in (34), where V_N^+ depends on A and C. Also, M_u depends on M_0, U_N, and T_N. U_N is given in (27) and (29) in terms of A and B. T_N in (28) and (30) depends on A, B, and C.

In the next step, it is shown how to use the structural dependence in (35) yet avoid knowledge of A, B, and C.

Define the vector of the observed data at time k as

$$\overline{z}_{k-1,k-N} = \begin{bmatrix} \overline{u}_{k-1,k-N} \\ \overline{y}_{k-1,k-N} \end{bmatrix}. \tag{42}$$

Now, one has

$$\begin{aligned} V^\mu(x_k) &= x_k^T P x_k \\ &= \overline{z}_{k-1,k-N}^T \begin{bmatrix} M_u^T \\ M_y^T \end{bmatrix} P [\, M_u \quad M_y \,] \overline{z}_{k-1,k-N} \end{aligned} \tag{43}$$

$$\begin{aligned} V^\mu(x_k) &= \overline{z}_{k-1,k-N}^T \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix} \overline{z}_{k-1,k-N} \\ &\equiv \overline{z}_{k-1,k-N}^T \bar{P} \overline{z}_{k-1,k-N}. \end{aligned} \tag{44}$$

Note that $\overline{u}_{k-1,k-N} \in R^{mN}$, $\overline{y}_{k-1,k-N} \in R^{pN}$, $\overline{z}_{k-1,k-N} \in R^{(m+p)N}$, and $\bar{P} \in R^{(m+p)N \times (m+p)N}$.

Equation (44) expresses the value function at time k as a quadratic form in terms of the past inputs and outputs.

Note that the inner kernel matrix $\bar{P}$ in (44) depends on the system dynamics A, B, and C. In the next section, it is shown how to use RL methods to learn the kernel matrix $\bar{P}$ without knowing A, B, and C.

B. Writing the TD Error in Terms of Available Measured Data

We may now write Bellman's equation (9) in terms of the observed data as

$$\begin{aligned} &\overline{z}_{k-1,k-N}^T \bar{P} \overline{z}_{k-1,k-N} \\ &\qquad = y_k^T Q y_k + u_k^T R u_k + \overline{z}_{k,k-N+1}^T \bar{P} \overline{z}_{k,k-N+1}. \end{aligned} \tag{45}$$

Based on this equation, write the temporal difference error (17) in terms of the inputs and outputs as

$$\begin{aligned} e_k = &-\overline{z}_{k-1,k-N}^T \bar{P} \overline{z}_{k-1,k-N} + y_k^T Q y_k + u_k^T R u_k \\ &+ \overline{z}_{k,k-N+1}^T \bar{P} \overline{z}_{k,k-N+1}. \end{aligned} \tag{46}$$

Using this TD error, the policy evaluation step of any form of RL based on the Bellman temporal difference error (17) can be equivalently performed using only the measured data, not the state.

Matrix $\bar{P}$ depends on A, B, and C through M_y and M_u. However, RL methods allow one to learn $\bar{P}$ online without A, B, C, as shown next.

C. Writing the Policy Update in Terms of Available Measured Data

Using Q-learning, one can perform PI and VI without any knowledge of the system dynamics. Likewise, it is now shown that, using the aforementioned constructions, one can derive a form for the policy improvement step (20)/(25) that does not depend on the state dynamics but only on the measured input/output data.

The policy improvement step may be written in terms of the observed data as

$$\mu(x_k) = \arg\min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \right) \tag{47}$$

$$\begin{aligned} \mu(x_k) = \arg\min_{u_k} \big(& y_k^T Q y_k + u_k^T R u_k \\ & + \overline{z}_{k,k-N+1}^T \bar{P} \overline{z}_{k,k-N+1} \big). \end{aligned} \tag{48}$$

Partition $\overline{z}_{k,k-N+1}^T \bar{P} \overline{z}_{k,k-N+1}$ as

$$\begin{aligned} &\overline{z}_{k,k-N+1}^T \bar{P} \overline{z}_{k,k-N+1} \\ &= \begin{bmatrix} u_k \\ \overline{u}_{k-1,k-N+1} \\ \overline{y}_{k,k-N+1} \end{bmatrix}^T \begin{bmatrix} p_0 & p_u & p_y \\ p_u^T & P_{22} & P_{23} \\ p_y^T & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} u_k \\ \overline{u}_{k-1,k-N+1} \\ \overline{y}_{k,k-N+1} \end{bmatrix}. \end{aligned} \tag{49}$$

One has $p_0 \in R^{m \times m}$, $p_u \in R^{m \times (m(N-1))}$, and $p_y \in R^{m \times pN}$. Then, differentiating with respect to u_k to perform the minimization in (48) yields

$$0 = R u_k + p_0 u_k + p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1} \tag{50}$$

or

$$u_k = -(R + p_0)^{-1} (p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1}). \tag{51}$$

This is a dynamic polynomial autoregressive moving-average (ARMA) controller that generates the current control input u_k in terms of the previous inputs and the current and previous outputs.

Exactly as in Q-learning (called by Werbos as *action-dependent* learning [41]), the control input appears in quadratic form (49), so that the minimization in (48) can be carried out in terms of the learned kernel matrix $\bar{P}$ without resorting to the system dynamics. However, since (49) contains present and

past values of the input and output, the result is a dynamical controller in polynomial ARMA form.

We have developed the following RL algorithms that use only the measured input/output data and do not require measurements of the full state vector x_k.

Algorithm 4—PI Algorithm Using OPFB

Select a stabilizing initial control policy $u_k^0 = \mu^0$. Then, for $j = 0, 1, \ldots$, perform until convergence:

1. **Policy Evaluation:** Solve for $\bar{P}^{j+1}$ such that

$$0 = -\overline{z}_{k-1,k-N}^T \bar{P}^{j+1} \overline{z}_{k-1,k-N} + y_k^T Q y_k + \left(u_k^j\right)^T R u_k^j + \overline{z}_{k,k-N+1}^T \bar{P}^{j+1} \overline{z}_{k,k-N+1}. \quad (52)$$

2. **Policy Improvement:** Partition $\bar{P}$ as in (49). Then, define the updated policy by

$$\begin{aligned} u_k^{j+1} &= \mu^{j+1}(x_k) \\ &= -\left(R + p_0^{j+1}\right)^{-1} \cdot \left(p_u^{j+1} \overline{u}_{k-1,k-N+1} + p_y^{j+1} \overline{y}_{k,k-N+1}\right). \end{aligned} \quad (53)$$

■

Algorithm 5—VI Algorithm Using OPFB

Select any initial control policy $u_k^0 = \mu^0$. Then, for $j = 0, 1, \ldots$, perform until convergence:

1. **Policy Evaluation:** Solve for $\bar{P}^{j+1}$ such that

$$\begin{aligned} &\overline{z}_{k-1,k-N}^T \bar{P}^{j+1} \overline{z}_{k-1,k-N} \\ &\quad = y_k^T Q y_k + \left(u_k^j\right)^T R u_k^j + \overline{z}_{k,k-N+1}^T \bar{P}^{j} \overline{z}_{k,k-N+1}. \end{aligned} \quad (54)$$

2. **Policy Improvement:** Partition $\bar{P}$ as in (49). Then, define the updated policy by

$$\begin{aligned} u_k^{j+1} &= u^{j+1}(x_k) \\ &= -\left(R + p_0^{j+1}\right)^{-1} \cdot \left(p_u^{j+1} \overline{u}_{k-1,k-N+1} + p_y^{j+1} \overline{y}_{k,k-N+1}\right). \end{aligned} \quad (55)$$

■

Remark 1: These algorithms do not require measurements of the internal state vector x_k. They only require measurements at each time step k of the utility $r_k = y_k^T Q y_k + u_k^T R u_k$, the inputs from time $k - N$ to time k, and the outputs from time $k - N$ to time k. The policy evaluation step may be implemented using standard methods such as batch LS or RLS (see the next section).

Remark 2: The control policy given by these algorithms in the form of (51), (53), and (55) is a dynamic ARMA regulator in terms of the past inputs and the current and past outputs. As such, it optimizes a polynomial square-of-sums cost function that is equivalent to the LQR sum-of-squares cost function (2). This polynomial cost function could be determined using the techniques in [17].

Remark 3: PI Algorithm 4 requires an initial stabilizing control policy, while VI Algorithm 5 does not. As such, VI is suitable for control of open-loop unstable systems.

Remark 4: The PI algorithm requires an initial stabilizing control policy $u_k^0 = \mu^0$ that is required to be a function only of the observable data. Suppose that one can find an initial stabilizing state feedback gain $u_k^0 = \mu^0(x_k) = -K^0 x_k$. Then, the equivalent stabilizing OPFB ARMA controller is easy to find and is given by

$$u_k^0 = \mu^0(x_k) = -K^0 x_k = -K^0 \begin{bmatrix} M_u & M_y \end{bmatrix} \begin{bmatrix} \overline{u}_{k-1,k-N} \\ \overline{y}_{k-1,k-N} \end{bmatrix}.$$

The next result shows that the controller (51) is unique.

Lemma 2: Define M_0 and M_1 according to (34). Then, the control sequence generated by (51) is independent of M_1 and depends only on M_0. Moreover, (51) is equivalent to

$$u_k = -(R + B^T P B)^{-1} (p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1}) \quad (56)$$

where p_u and p_y depend only on M_0.

Proof: Write (50) as

$$\begin{aligned} 0 &= R u_k + p_0 u_k + p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1} \\ &= R u_k + \begin{bmatrix} p_0 & p_u \,|\, p_y \end{bmatrix} \begin{bmatrix} \overline{u}_{k,k-N+1} \\ \overline{y}_{k,k-N+1} \end{bmatrix}. \end{aligned}$$

According to (44) and (49)

$$\begin{bmatrix} p_0 & p_u \end{bmatrix} = \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T P M_u \quad p_y \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T P M_y$$

so

$$\begin{aligned} 0 &= R u_k + p_0 u_k + p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1} \\ &= R u_k + \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T P (M_u \overline{u}_{k,k-N+1} + M_y \overline{y}_{k,k-N+1}). \end{aligned}$$

According to Lemma 1, this is unique, independently of M_1 [see (38)], and equal to

$$\begin{aligned} 0 = R u_k + \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T \\ \times P \left((U_N - M_0 T_N) \overline{u}_{k,k-N+1} + M_0 \overline{y}_{k,k-N+1}\right). \end{aligned}$$

Now

$$\begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T P = \left(M_u \begin{bmatrix} I_m \\ 0 \end{bmatrix}\right)^T P.$$

However

$$M_u \begin{bmatrix} I_m \\ 0 \end{bmatrix} = (U_N - (M_0 + M_1) T_N) \begin{bmatrix} I_m \\ 0 \end{bmatrix} = B$$

where one has used the structure of U_N and T_N. Consequently

$$\begin{aligned} 0 &= R u_k + p_0 u_k + p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1} \\ &= R u_k + B^T P \left((U_N - M_0 T_N) \overline{u}_{k,k-N+1} + M_0 \overline{y}_{k,k-N+1}\right) \end{aligned}$$

which is independent of M_1.

Now note that $p_0 = [\, I_m \quad 0\,] M_u^T P M_u \begin{bmatrix} I_m \\ 0 \end{bmatrix} = B^T P B$ as per the previous one. Hence, (56) follows from (51). ■

Remark 5: Note that the controller cannot be implemented in the form of (56) since it requires that matrix P be known.

IV. Implementation, Probing Noise, Bias, and Discount Factors

In this section, we discuss the need for probing noise to implement the aforementioned algorithms, show that this noise leads to deleterious effects such as bias, and argue how adding a discount factor in the cost (2) can reduce this bias to a negligible effect.

The equations in PI and VI are solved online by standard techniques using methods such as batch LS or RLS. See [5], where RLS was used in Q-learning. In PI, one solves (52) by writing it in the form of

$$stk(\bar{P}^{j+1})\,[\overline{z}_{k-1,k-N} \otimes \overline{z}_{k-1,k-N} - \overline{z}_{k,k-N+1} \otimes \overline{z}_{k,k-N+1}] = y_k^T Q y_k + \left(u_k^j\right)^T R u_k^j \quad (57)$$

with $\otimes$ being the Kronecker product and $stk(.)$ being the column stacking operator [6]. The redundant quadratic terms in the Kronecker product are combined. In VI, one solves (54) in the form of

$$stk(\bar{P}^{j+1})\,[\overline{z}_{k-1,k-N} \otimes \overline{z}_{k-1,k-N}] = y_k^T Q y_k + \left(u_k^j\right)^T R u_k^j + \overline{z}_{k,k-N+1}^T \bar{P}^j \overline{z}_{k,k-N+1}. \quad (58)$$

Both of these equations only require that the input/output data be measured. They are scalar equations, yet one must solve for the kernel matrix $\bar{P} \in R^{(m+p)N \times (m+p)N}$, which is symmetric and has $[(m+p)N][(m+p)N+1]/2$ independent terms. Therefore, one requires data samples for at least $[(m+p)N][(m+p)N+1]/2$ time steps for a solution using batch LS.

To solve the PI update equation (57), it is required that the quadratic vector $[\overline{z}_{k-1,k-N} \otimes \overline{z}_{k-1,k-N} - \overline{z}_{k,k-N+1} \otimes \overline{z}_{k,k-N+1}]$ be linearly independent over time, which is a property known as persistence of excitation (PE). To solve the VI update equation (58), one requires the PE of the quadratic vector $[\overline{z}_{k-1,k-N} \otimes \overline{z}_{k-1,k-N}]$. It is standard practice (see [5] for instance) to inject probing noise into the control action to obtain PE, so that one puts into the system dynamics the input $\hat{u}_k = u_k + d_k$, with u_k being the control computed by the PI or VI current policy and d_k being a probing noise or dither, e.g., white noise.

It is well known that dither can cause biased results and mismatch in system identification. In [44], this issue is discussed, and several alternative methods are presented for injecting dither into system identification schemes to obtain improved results. Unfortunately, in control applications, one has little choice about where to inject the probing noise.

To see the deleterious effects of probing noise, consider the Bellman equation (9) with input $\hat{u}_k = u_k + d_k$, where d_k is a probing noise. One writes

$$\begin{aligned} x_k^T \hat{P} x_k &= y_k^T Q y_k + \hat{u}_k^T R \hat{u}_k + \hat{x}_{k+1}^T \hat{P} \hat{x}_{k+1} \\ x_k^T \hat{P} x_k &= y_k^T Q y_k + (u_k + d_k)^T R (u_k + d_k) \\ &\quad + (Ax_k + Bu_k + Bd_k)^T \hat{P} (Ax_k + Bu_k + Bd_k) \\ x_k^T \hat{P} x_k &= y_k^T Q y_k + u_k^T R u_k + d_k^T R d_k + u_k^T R d_k \\ &\quad + d_k^T R u_k + (Ax_k + Bu_k)^T \hat{P} (Ax_k + Bu_k) \\ &\quad + (Ax_k + Bu_k)^T \hat{P} B d_k + (Bd_k)^T \hat{P} (Ax_k + Bu_k) \\ &\quad + (Bd_k)^T \hat{P} B d_k. \end{aligned}$$

Now, use $tr\{AB\} = tr\{BA\}$ for commensurate matrices A and B, take the expected values to evaluate correlation matrices, and assume that the dither at time k is white noise independent of u_k and x_k so that $E\{Ru_k d_k^T\} = 0$ and $E\{\hat{P} B d_k (Ax_k + Bu_k)^T\} = 0$ and that the cross terms drop out. Then, averaged over repeated control runs with different probing noise sequence d_k, this equation is effectively

$$x_k^T \hat{P} x_k = y_k^T Q y_k + u_k^T R u_k + (Ax_k + Bu_k)^T \hat{P} (Ax_k + Bu_k) + d_k^T (B^T \hat{P} B + R) d_k$$

which is the undithered Bellman equation plus a term depending on the dither covariance. As such, the solution computed by PI or VI will not correspond to the actual value corresponding to the Bellman equation.

It is now argued that discounting the cost can significantly reduce the deleterious effects of probing noise. Adding a discount factor $\gamma < 1$ to the cost (2) results in

$$V^{\mu}(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} \left(y_i^T Q y_i + u_i^T R u_i\right) \equiv \sum_{i=k}^{\infty} \gamma^{i-k} r_i \quad (59)$$

with the associated Bellman equation

$$x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + \gamma x_{k+1}^T P x_{k+1}. \quad (60)$$

This has an HJB (Riccati) equation that is equivalent to

$$0 = -P + \gamma \left(A^T P A - A^T P B (R/\gamma + B^T P B)^{-1} B^T P A\right) + C^T Q C \quad (61)$$

and an optimal policy of

$$u_k = -(R/\gamma + B^T P B)^{-1} B^T P A x_k. \quad (62)$$

The benefits of discounting are most clearly seen by examining VI. VI Algorithm 3, which, with discount, has (24) modified as

$$x_k^T P^{j+1} x_k = y_k^T Q y_k + \left(u_k^j\right)^T R u_k^j + \gamma x_{k+1}^T P^j x_{k+1} \quad (63)$$

corresponds to the underlying Riccati difference equation

$$P_{j+1} = \gamma \left(A^T P_j A - A^T P_j B (R/\gamma + B^T P_j B)^{-1} B^T P_j A\right) + C^T Q C \quad (64)$$

where each iteration is decayed by the factor $\gamma < 1$. The effects of this can be best understood by considering the Lyapunov difference equation

$$P_{j+1} = \gamma A^T P_j A + C^T Q C, \qquad P_0 \tag{65}$$

which has a solution

$$P_j = \gamma^j (A^T)^j P_0 A^j + \sum_{i=0}^{j-1} \gamma^i (A^T)^i Q A^i. \tag{66}$$

The effect of the discount factor is thus to decay the effects of the initial conditions.

In a similar vein, the discount factor decays the effects of the previous probing noises and improper initial conditions in the PI and VI algorithms. In fact, adding a discount factor is closely related to adding exponential data weighting in the Kalman filter to remove the bias effects of unmodeled dynamics [18].

V. Simulations

A. Stable Linear System and PI

Consider the stable linear system with quadratic cost function

$$x_{k+1} = \begin{bmatrix} 1.1 & -0.3 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k$$

$$y_k = \begin{bmatrix} 1 & -0.8 \end{bmatrix} x_k$$

where Q and R in the cost function are the identity matrices of appropriate dimensions. The open-loop poles are $z_1 = 0.5$ and $z_1 = 0.6$. In order to verify the correctness of the proposed algorithm, the optimal value kernel matrix P is found by solving (15) to be

$$P = \begin{bmatrix} 1.0150 & -0.8150 \\ -0.8150 & 0.6552 \end{bmatrix}.$$

Now, by using $\bar{P} \equiv \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix}$ and (35), one has

$$\bar{P} = \begin{bmatrix} 1.0150 & -0.8440 & 1.1455 & -0.3165 \\ -0.8440 & 0.7918 & -1.0341 & 0.2969 \\ 1.1455 & -1.0341 & 1.3667 & -0.3878 \\ -0.3165 & 0.2969 & -0.3878 & 0.1113 \end{bmatrix}.$$

Since the system is stable, we use the OPFB PI Algorithm 4 implemented as in (52) and (53). PE was ensured by adding dithering noise to the control input, and a discount (forgetting) factor $\gamma = 0.2$ was added to diminish the dither bias effects. The observer index is $K = 2$, and N is selected to be equal to two.

By applying dynamic OPFB control (53), the system remained stable, and the parameters of $\bar{P}$ converged to the optimal ones. In fact

$$\hat{\bar{P}} = \begin{bmatrix} 1.1340 & -0.8643 & 1.1571 & -0.3161 \\ -0.8643 & 0.7942 & -1.0348 & 0.2966 \\ 1.1571 & -1.0348 & 1.3609 & -0.3850 \\ -0.3161 & 0.2966 & -0.3850 & 0.1102 \end{bmatrix}.$$

In the example, batch LS was used to solve (52) at each step.

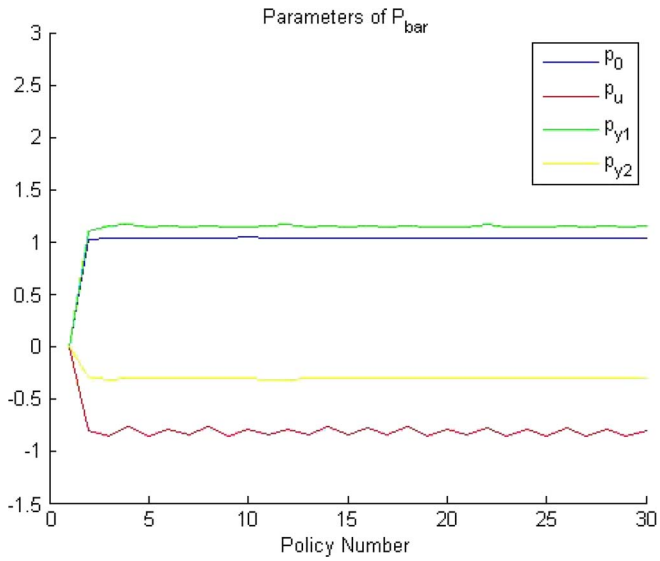

Fig. 1. Convergence of p_0, p_u, and p_y.

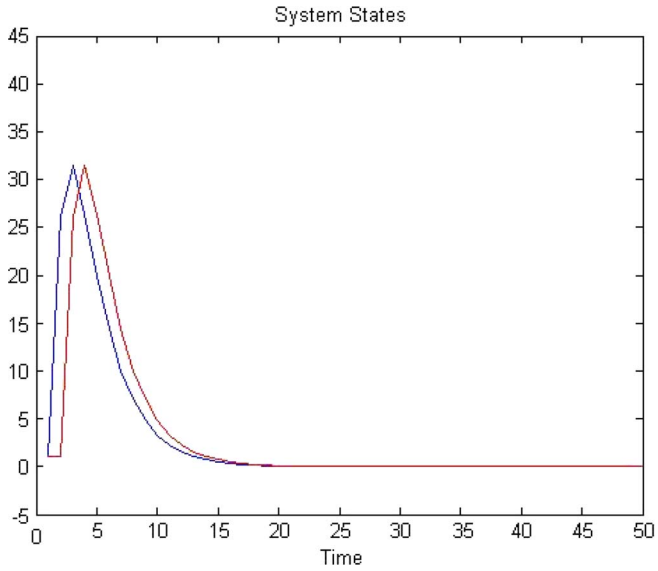

Fig. 2. Evolution of the system states for the duration of the experiment.

Fig. 1 shows the convergence of $p_0 \in R$, $p_u \in R$, and $p_y \in R^{1\times 2}$ of $\bar{P}$ to the correct values. Fig. 2 shows the evolution of the system states and their convergence to zero.

B. Q-Learning and OPFB ADP

The purpose of this example is to compare the performance of Q-learning [5], [39] and that of OPFB ADP. Consider the stable linear system described before, and apply Q-learning. The Q-function for this particular system is given by

$$\begin{aligned} Q_h(x_k, u_k) &= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T P A & A^T P B \\ B^T P A & R + B^T P B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ &\equiv \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ &= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} 4.9768 & -0.8615 & 2.8716 \\ -0.8615 & 1.2534 & -0.8447 \\ 2.8716 & -0.8447 & 3.8158 \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \end{aligned}$$

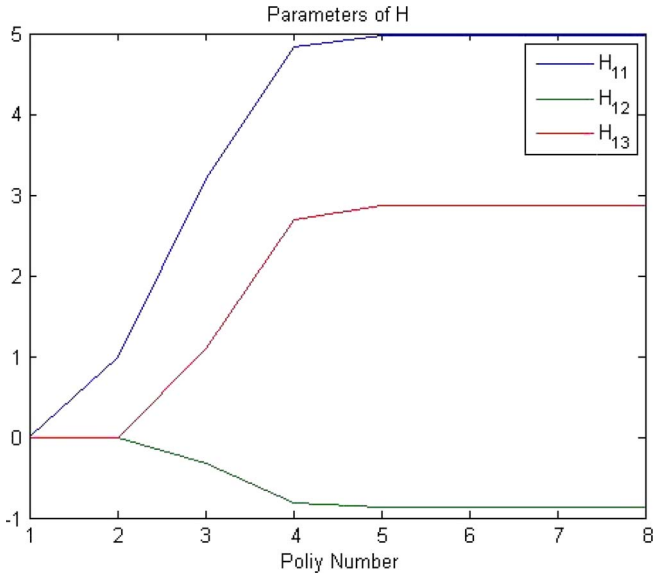

Fig. 3. Convergence of H_{11}, H_{12}, and H_{13}.

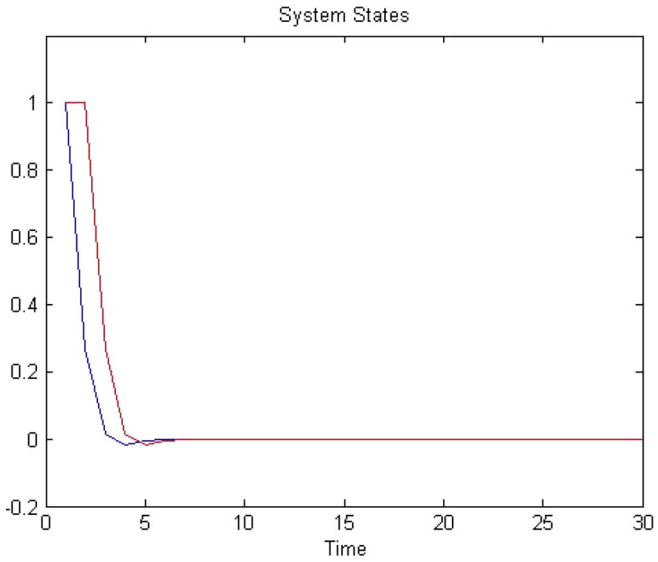

Fig. 4. Evolution of the system states for the duration of the experiment.

By comparing these two methods, it is obvious that Q-learning has a faster performance than OPFB ADP since fewer parameters are being identified.

Fig. 3 shows the convergence of the three parameters H_{11}, H_{12}, and H_{13} of the H matrix shown before to the correct values. Fig. 4 shows the evolution of the system states.

C. Unstable Linear System and VI

Consider the unstable linear system with quadratic cost function

$$x_{k+1} = \begin{bmatrix} 1.8 & -0.7700 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k$$

$$y_k = \begin{bmatrix} 1 & -0.5 \end{bmatrix} x_k$$

where Q and R in the cost function are the identity matrices of appropriate dimensions. The open-loop poles are $z_1 = 0.7$ and $z_2 = 1.1$, so the system is unstable. In order to verify the correctness of the proposed algorithm, the optimal value kernel matrix P is found by solving (15) to be

$$P = \begin{bmatrix} 1.3442 & -0.7078 \\ -0.7078 & 0.3756 \end{bmatrix}.$$

Now, by using $\bar{P} \equiv \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix}$ and (35), one has

$$\bar{P} = \begin{bmatrix} 1.3442 & -0.7465 & 2.4582 & -1.1496 \\ -0.7465 & 0.4271 & -1.3717 & 0.6578 \\ 2.4582 & -1.3717 & 4.4990 & -2.1124 \\ -1.1496 & 0.6578 & -2.1124 & 1.0130 \end{bmatrix}.$$

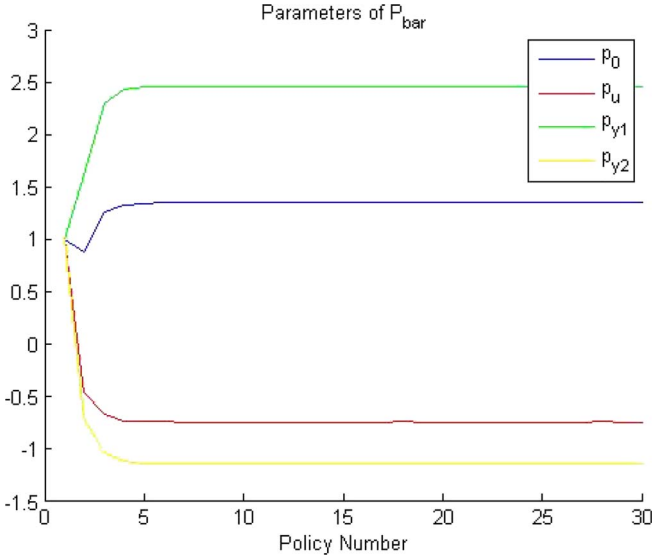

Fig. 5. Convergence of p_0, p_u, and p_y.

Since the system is open-loop unstable, one must use VI, not PI. OPFB VI Algorithm 5 is implemented as in (54) and (55). PE was ensured by adding dithering noise to the control input, and a discount (forgetting) factor $\gamma = 0.2$ was used to diminish the dither bias effects. The observer index is $K = 2$, and N is selected to be equal to two.

By applying dynamic OPFB control (55), the system remained stabilized, and the parameters of $\bar{P}$ converged to the optimal ones. In fact

$$\hat{\bar{P}} = \begin{bmatrix} 1.3431 & -0.7504 & 2.4568 & -1.1493 \\ -0.7504 & 0.4301 & -1.3730 & 0.6591 \\ 2.4568 & -1.3730 & 4.4979 & -2.1120 \\ -1.1493 & 0.6591 & -2.1120 & 1.0134 \end{bmatrix}.$$

In the example, batch LS was used to solve (54) at each step. Fig. 5 shows the convergence of $p_0 \in R$, $p_u \in R$, and $p_y \in R^{1\times 2}$ of $\bar{P}$. Fig. 6 shows the evolution of the system states, their boundedness despite the fact that the plant is initially unstable, and their convergence to zero.

VI. Conclusion

In this paper, we have proposed the implementation of ADP using only the measured input/output data from a dynamical system. This is known in control system theory as "OPFB," as

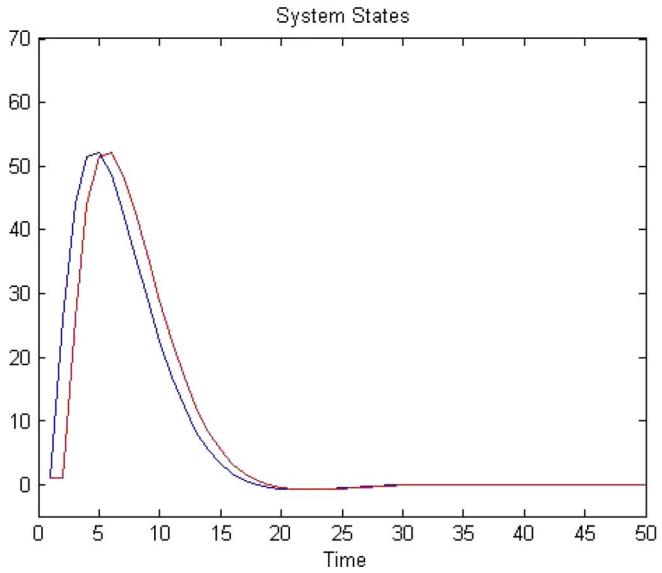

Fig. 6. Evolution of the system states for the duration of the experiment.

opposed to full state feedback, and corresponds to RL for a class of POMDPs. Both PI and VI algorithms have been developed that require only OPFB. An added and surprising benefit is that, similar to Q-learning, the system dynamics are not needed to implement these OPFB algorithms, so that they converge to the optimal controller for completely unknown systems. The system order, as well as an upper bound on its "observability index," must be known. The learned OPFB controller is given in the form of a polynomial ARMA controller that is equivalent to the optimal state variable feedback gain. This controller needs the addition of probing noise in order to be sufficiently rich. This probing noise adds some bias, and in order to avoid it, a discount factor is added in the cost.

Future research efforts will focus on how to apply the previous method to nonlinear systems.

REFERENCES

[1] W. Aangenent, D. Kostic, B. de Jager, R. van de Molengraft, and M. Steinbuch, "Data-based optimal control," in *Proc. Amer. Control Conf.*, Portland, OR, 2005, pp. 1460–1465.

[2] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 943–949, Aug. 2008.

[3] J. K. Bennighof, S. M. Chang, and M. Subramaniam, "Minimum time pulse response based control of flexible structures," *J. Guid. Control Dyn.*, vol. 16, no. 5, pp. 874–881, 1993.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[5] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, "Adaptive linear quadratic control using policy iteration," in *Proc. Amer. Control Conf.*, Baltimore, MD, Jun. 1994, pp. 3475–3476.

[6] J. W. Brewer, "Kronecker products and matrix calculus in system theory," *IEEE Trans. Circuits Syst.*, vol. CAS-25, no. 9, pp. 772–781, Sep. 1978.

[7] X. Cao, *Stochastic Learning and Optimization*. Berlin, Germany: Springer-Verlag, 2007.

[8] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput.*, vol. 12, no. 1, pp. 219–245, Jan. 2000.

[9] K. Doya, H. Kimura, and M. Kawato, "Neural mechanisms of learning and control," *IEEE Control Syst. Mag.*, vol. 21, no. 4, pp. 42–54, Aug. 2001.

[10] G. Hewer, "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," *IEEE Trans. Autom. Control*, vol. AC-16, no. 4, pp. 382–384, Aug. 1971.

[11] H. Hjalmarsson and M. Gevers, "Iterative feedback tuning: Theory and applications," *IEEE Control Syst. Mag.*, vol. 18, no. 4, pp. 26–41, Aug. 1998.

[12] P. Ioannou and B. Fidan, "Advances in design and control," in *Adaptive Control Tutorial*. Philadelphia, PA: SIAM, 2006.

[13] M. Krstic and H. Deng, *Stabilization of Nonlinear Uncertain Systems*. Berlin, Germany: Springer-Verlag, 1998.

[14] P. Lancaster and L. Rodman, *Algebraic Riccati Equations (Oxford Science Publications)*. New York: Oxford Univ. Press, 1995.

[15] A. Lecchini, M. C. Campi, and S. M. Savaresi, "Virtual reference feedback tuning for two degree of freedom controllers," *Int. J. Adapt. Control Signal Process.*, vol. 16, no. 5, pp. 355–371, 2002.

[16] F. L. Lewis, G. Lendaris, and D. Liu, "Special issue on approximate dynamic programming and reinforcement learning for feedback control," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 896–897, Aug. 2008.

[17] F. L. Lewis and V. L. Syrmos, *Optimal Control*. New York: Wiley, 1995.

[18] F. L. Lewis, L. Xie, and D. O. Popa, *Optimal and Robust Estimation*. Boca Raton, FL: CRC Press, Sep. 2007.

[19] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, Sep. 2009.

[20] Z. H. Li and M. Krstic, "Optimal design of adaptive tracking controllers for nonlinear systems," in *Proc. ACC*, 1997, pp. 1191–1197.

[21] R. K. Lim, M. Q. Phan, and R. W. Longman, "State-Space System Identification with Identified Hankel Matrix," Dept. Mech. Aerosp. Eng., Princeton Univ., Princeton, NJ, Tech. Rep. 3045, Sep. 1998.

[22] R. K. Lim and M. Q. Phan, "Identification of a multistep-ahead observer and its application to predictive control," *J. Guid. Control Dyn.*, vol. 20, no. 6, pp. 1200–1206, 1997.

[23] preprint P. Mehta and S. Meyn, Q-learning and Pontryagin's minimum principle2009.

[24] W. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York: Wiley, 2007.

[25] D. Prokhorov and D. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.

[26] M. Q. Phan, R. K. Lim, and R. W. Longman, "Unifying Input–Output and State-Space Perspectives of Predictive Control," Dept. Mech. Aerosp. Eng., Princeton Univ., Princeton, NJ, Tech. Rep. 3044, Sep. 1998.

[27] M. G. Safonov and T. C. Tsao, "The unfalsified control concept and learning," *IEEE Trans. Autom. Control*, vol. 42, no. 6, pp. 843–847, Jun. 1997.

[28] W. Schultz, "Neural coding of basic reward terms of animal learning theory, game theory, microeconomics and behavioral ecology," *Curr. Opin. Neurobiol.*, vol. 14, no. 2, pp. 139–147, Apr. 2004.

[29] W. Schultz, P. Dayan, and P. Read Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, no. 5306, pp. 1593–1599, Mar. 1997.

[30] J. Si, A. Barto, W. Powel, and D. Wunch, *Handbook of Learning and Approximate Dynamic Programming*. Englewood Cliffs, NJ: Wiley, 2004.

[31] R. E. Skelton and G. Shi, "Markov Data-Based LQG Control," *Trans. ASME, J. Dyn. Syst. Meas. Control*, vol. 122, no. 3, pp. 551–559, 2000.

[32] J. C. Spall and J. A. Criston, "Model-free control of nonlinear stochastic systems with discrete-time measurements," *IEEE Trans. Autom. Control*, vol. 43, no. 9, pp. 1198–1210, Sep. 1998.

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning—An Introduction*. Cambridge, MA: MIT Press, 1998.

[34] R. L. Toussaint, J. C. Boissy, M. L. Norg, M. Steinbuch, and O. H. Bosgra, "Suppressing non-periodically repeating disturbances in mechanical servo systems," in *Proc. IEEE Conf. Decision Control*, Tampa, FL, 1998, pp. 2541–2542.

[35] D. Vrabie, K. Vamvoudakis, and F. Lewis, "Adaptive optimal controllers based on generalized policy iteration in a continuous-time framework," in *Proc. IEEE Mediterranean Conf. Control Autom.*, Jun. 2009, pp. 1402–1409.

[36] K. G. Vamvoudakis and F. L. Lewis, "Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem," in *Proc. Int. Joint Conf. Neural Netw.*, Atlanta, GA, Jun. 2009, pp. 3180–3187.

[37] K. Vamvoudakis, D. Vrabie, and F. L. Lewis, "Online policy iteration based algorithms to solve the continuous-time infinite horizon optimal control problem," in *Proc. IEEE Symp. ADPRL*, Nashville, TN, Mar. 2009, pp. 36–41.

[38] F. Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 39–47, May 2009.

[39] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge Univ., Cambridge, U.K., 1989.

[40] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavior sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA, 1974.
[41] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control*, D. A. White and D. A. Sofge, Eds. New York: Van Nostrand Reinhold, 1992.
[42] P. Werbos, "Neural networks for control and system identification," in *Proc. IEEE CDC*, 1989, pp. 260–265.
[43] D. A. White and D. A Sofge, Eds., *Handbook of Intelligent Control*. New York: Van Nostrand Reinhold, 1992.
[44] B. Widrow and E. Walach, *Adaptive Inverse Control*. Upper Saddle River, NJ: Prentice-Hall, 1996.

F. L. Lewis (S'78–M'81–SM'86–F'94) received the B.S. degree in physics/electrical engineering and the M.S.E.E. degree from Rice University, Houston, TX, the M.S. degree in aeronautical engineering from the University of West Florida, Pensacola, and the Ph.D. degree from the Georgia Institute of Technology, Atlanta.

He is currently a Distinguished Scholar Professor and the Moncrief-O'Donnell Chair with the Automation and Robotics Research Institute (ARRI), The University of Texas at Arlington, Fort Worth. He works in feedback control, intelligent systems, distributed control systems, and sensor networks. He is the author of 216 journal papers, 330 conference papers, 14 books, 44 chapters, and 11 journal special issues and is the holder of six U.S. patents.

Dr. Lewis is a Fellow of the International Federation of Automatic Control and the U.K. Institute of Measurement and Control, a Professional Engineer in Texas, and a U.K. Chartered Engineer. He was the Founding Member of the Board of Governors of the Mediterranean Control Association. He served on the National Academy of Engineering Committee on Space Station in 1995. He is an elected Guest Consulting Professor with both South China University of Technology, Guangzhou, China, and Shanghai Jiao Tong University, Shanghai, China. He received the Fulbright Research Award, the National Science Foundation Research Initiation Grant, the American Society for Engineering Education Terman Award, the 2009 International Neural Network Society Gabor Award, the U.K. Institute of Measurement and Control Honeywell Field Engineering Medal in 2009, and the Outstanding Service Award from the Dallas IEEE Section. He was selected as Engineer of the Year by the Fort Worth IEEE Section and listed in *Fort Worth Business Press Top 200 Leaders in Manufacturing*. He helped win the IEEE Control Systems Society Best Chapter Award (as the Founding Chairman of the Dallas fort Worth, Chapter), the National Sigma Xi Award for Outstanding Chapter (as the President of the University of Texas at Arlington Chapter), and the U.S. Small business Administration Tibbets Award in 1996 (as the Director of ARRI's Small Business Innovation Research Program).

Kyriakos G. Vamvoudakis (S'02–M'06) was born in Athens, Greece. He received the Diploma in Electronic and Computer Engineering (with highest honors) from the Technical University of Crete, Chania, Greece, in 2006 and the M.Sc. degree in electrical engineering from The University of Texas at Arlington, in 2008, where he is currently working toward the Ph.D. degree.

He is also currently a Research Assistant with the Automation and Robotics Research Institute, The University of Texas at Arlington. His current research interests include approximate dynamic programming, neural-network feedback control, optimal control, adaptive control, and systems biology.

Mr. Vamvoudakis is a member of Tau Beta Pi, Eta Kappa Nu, and Golden Key honor societies and is listed in *Who's Who in the World*. He is a Registered Electrical/Computer Engineer (Professional Engineer) and a member of the Technical Chamber of Greece.