Differential Temporal Difference Learning

Adithya M. Devraj, Ioannis Kontoyiannis, Fellow, IEEE, and Sean P. Meyn, Fellow, IEEE

Abstract—Value functions derived from Markov decision processes arise as a central component of algorithms as well as performance metrics in many statistics and engineering applications of machine learning. Computation of the solution to the associated Bellman equations is challenging in most practical cases of interest. A popular class of approximation techniques, known as temporal difference (TD) learning algorithms, are an important subclass of general reinforcement learning methods. The algorithms introduced in this article are intended to resolve two well-known issues with TD-learning algorithms. Their slow convergence due to very high central limit theorem variance, and the fact that, for the problem of computing the relative value function, consistent algorithms exist only in special cases. First we show that the gradients of these value functions admit a representation that lends itself to algorithm design. Based on this result, a new class of differential TD-learning algorithms is introduced. For Markovian models on Euclidean space with smooth dynamics, the algorithms are shown to be consistent under general conditions. Numerical results show dramatic variance reduction in comparison to standard methods.

Index Terms—Approximate dynamic programming, Poisson equation, reinforcement learning, stochastic optimal control, temporal difference (TD) learning.

I. INTRODUCTION

A CENTRAL task in the application of many machine learning methods and control techniques is the (exact or approximate) computation of value functions arising from Markov decision processes. The class of temporal difference (TD) learning algorithms considered in this article is an important subclass of the general family of reinforcement learning methods that performs this task. Our main contributions here are the introduction of a related family of TD-learning algorithms that enjoy better convergence properties than existing methods, and the rigorous theoretical analysis of these algorithms.

The value functions considered in this article are based on a discrete-time Markov chain $X = \{X(t) : t = 0, 1, 2, \ldots\}$ taking values in $\mathbb{R}^d$, and on an associated cost function $c : \mathbb{R}^d \to \mathbb{R}$. Our central modeling assumption throughout is that $X$ evolves according to the nonlinear state space model

$$X(t+1) = a(X(t), N(t+1)), \quad t \geq 0$$

where $N = \{N(t) : t = 0, 1, 2, \ldots\}$ is an $m$-dimensional disturbance sequence of independent and identically distributed (i.i.d.) random variables, and $a : \mathbb{R}^{d+m} \to \mathbb{R}^d$ is a continuous mapping.

Under these assumptions, for all $t \geq 0$, $X(t+1)$ is a continuous function of the initial condition $X(0) = x$; this observation is our starting point for the construction of effective algorithms for value function approximation.

We begin with some familiar background.

A. Value Functions

Given a discount factor $\beta \in (0, 1)$, the discounted-cost value function is defined as

$$h_\beta(x) := \sum_{t=0}^{\infty} \beta^t \mathbb{E}[c(X(t)) \mid X(0) = x], \quad x \in \mathbb{R}^d.$$  (2)

It is known that $h_\beta$ solves the Bellman equation [2], [3]

$$c(x) + \beta h_\beta(X(t+1)) - h_\beta(x) = 0.$$  (3)

The average cost is defined as the ergodic limit

$$\eta := \lim_{n \to \infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathbb{E}[c(X(t)) \mid X(0) = x]$$

where the limit exists and is independent of $x$ under the conditions discussed in Section II. The following relative value function is central to analysis of average cost control problems:

$$h(x) := \sum_{t=0}^{\infty} \mathbb{E}[c(X(t)) - \eta \mid X(0) = x], \quad x \in \mathbb{R}^d.$$  (5)

Provided the sum (5) exists for each $x$, the relative value function solves the Poisson equation [6], [3]

$$\mathbb{E}[h(X(t+1)) \mid X(t) = x] - h(x) = -[c(x) - \eta].$$  (6)
These equations and their solutions are of interest in learning theory, control engineering, and many other fields, which include the following.

1) **Optimal control and Markov decision processes:** Policy iteration and actor-critic algorithms are designed to approximate an optimal policy using two-step procedures: First, given a policy, the associated value function is computed (or approximated), and then the policy is updated based on this value function [7], [8]. These approaches can be used for both discounted- and average-cost optimal control problems.

2) **Algorithm design for variance reduction:** Under general conditions, the asymptotic variance [i.e., the variance appearing in the central limit theorem for the averages in (4)] is naturally expressed in terms of the relative value function $h$ [9], [10]. The method of control variate is intended to reduce the asymptotic variance of various Monte Carlo methods; a version of this technique involves the construction of an approximation to $h$ [11]–[15].

3) **Nonlinear filtering:** A recent approach to approximate nonlinear filtering requires the gradient of the solution to Poisson’s equation to obtain the “innovation gain” [16], [17]. Approximations of the solution can lead to efficient implementations of this method [18]–[20].

**B. TD-Learning and Value Function Approximation**

In most cases of practical interest, closed-form expressions for the value functions $h_\beta$ and $h$ in (2) or (6) cannot be derived. One approach to obtaining approximations is the TD-iteration algorithm [2], [21].

In the case of the discounted-cost value function, the goal of TD-learning is to approximate $h_\beta$ as a member of a parametrized family of functions $\{h_\beta^\theta : \theta \in \mathbb{R}^d\}$. Throughout, we restrict attention to linear parametrizations of the form

$$h_\beta^\theta = \sum_{j=1}^d \theta_j \psi_j$$

(7)

where we write $\theta = (\theta_1, \theta_2, \ldots, \theta_d)^T$, $\psi = (\psi_1, \psi_2, \ldots, \psi_d)^T$, and we assume that the given collection of “basis” functions $\psi: \mathbb{R}^d \rightarrow \mathbb{R}$ is continuously differentiable.

For any function $f: \mathbb{R}^d \rightarrow \mathbb{R}$ and a given probability measure $\mu: \mathbb{R}^d \rightarrow [0, 1]$, we define the $\mu$-norm

$$\|f\|_\mu := \left(\mathbb{E}[f^2(X)]\right)^{\frac{1}{2}}, \quad X \sim \mu$$

$$= \left(\int f^2(x) \mu(dx)\right)^{\frac{1}{2}}.$$  

In one variant of the TD technique [the least squares TD (LSTD) (1) algorithm], described in Section IV, the optimal parameter vector $\theta^*$ is chosen as the solution to a minimum-norm problem

$$\theta^* = \arg \min_{\theta} \|h_\beta^\theta - h_\beta\|_2^2$$

(8)

where the expectation is with respect to $X \sim \pi$, and $\pi$ denotes the steady-state distribution of the Markov chain $X$; more details are provided in Sections II-A and IV.

A TD-learning algorithm is said to be consistent, if the parameter estimates obtained using the algorithm converge to $\theta^*$.

Theory for TD-learning in the discounted-cost setting is largely complete, in the sense that criteria for convergence are well-understood, and the asymptotic variance of the algorithm is computable based on standard theory from stochastic approximation [22], [23], [25] (see [4], [5], [24], for the relationship between asymptotic variance and convergence rate of TD-learning algorithms). Theory and algorithms for the average-cost setting involving the relative value function $h$ is more fragmented. The optimal parameter $\theta^*$ in the analog of (8) with $h_\beta$ replaced by the relative value function $h$ can be computed using TD-learning techniques only for Markovian models that regenerate: there exists a state $x^* \in \mathbb{R}^d$ that is visited infinitely often (see the regenerative TD(1) algorithm in [43, p. 1012], and also [3], [26]).

Regeneration is often not a restrictive assumption. However, the asymptotic variance of these algorithms grows with the variance of inter-regeneration times (time between consecutive visits to the regenerative state $x^*$). The variance can be massive even in simple examples such as the M/M/1 queue; see the final chapter of [3]. High variance is also predominantly observed in the discounted-cost case when the discounting factor is close to 1 (see the relevant remarks in Section I-D).

The differential TD-learning algorithms developed in this article are designed in part to resolve these issues. The main idea is to estimate the gradient of the value function. Under the conditions imposed, the asymptotic variance of the resulting algorithms remains uniformly bounded over $0 < \beta < 1$. And the same techniques can be applied to obtain finite-variance algorithms for approximating the relative value function $h$ for models without regeneration.

We note that the needs of the analysis of the algorithms presented here have, in part, motivated the development of rich new convergence theory for general classes of Markov processes [28]. Indeed, the results in Sections II and III draw heavily on the convergence results established in [28].

**C. Differential TD-Learning**

Consider the discounted-cost setting: suppose that the value function $h_\beta$ and all its potential approximations $\{h_\beta^\theta : \theta \in \mathbb{R}^d\}$ are continuously differentiable as functions of the state $x$, i.e., $h_\beta, h_\beta^\theta \in C^2$, for each $\theta \in \mathbb{R}^d$. In terms of the linear parameterization (7), we obtain approximations of the form

$$\nabla h_\beta^\theta = \sum_{j=1}^d \theta_j \nabla \psi_j$$

(9)

where the gradient is with respect to the state $x$.

The differential LSTD-learning algorithm introduced in Section III is designed to compute the solution to

$$\theta^* = \arg \min_{\theta} \mathbb{E}[\|\nabla h_\beta^\theta(X) - \nabla h_\beta(X)\|_2^2], \quad X \sim \pi$$

(10)

where $\|\cdot\|_2$ is the usual Euclidean norm, and once again, $\pi$ denotes the steady-state distribution of the Markov chain $X$. 
The value function approximation \( h_\beta^\theta \) is obtained via the addition of a constant

\[
    h_\beta^\theta = \sum_{j=1}^{d} \theta_j^* \phi_j + \kappa(\theta^*). \tag{11}
\]

The mean-square optimal choice is obtained on requiring

\[
    E[h_\beta^\theta(X) - h_\beta(X)] = 0, \quad X \sim \pi. \tag{12}
\]

See the discussion that follows Algorithm 2 for details.

A similar program can be carried out for the relative value function \( h \), which, viewed as a solution to Poisson’s equation (6), is unique only up to an additive constant. Therefore, we can set \( \kappa(\theta^*) = 0 \) in the average-cost setting.

**D. Summary of Contributions**

The main contributions of this article are the following.

1) The introduction of the new differential least squares TD-learning (\( \nabla \)-LSTD, or “grad-LSTD”) algorithm, which is applicable in both the discounted- and average-cost settings.

2) The development of appropriate conditions under which we can show that, for linear parameterization, \( \nabla \)-LSTD converges, and solve the quadratic program (10).

3) The introduction of the family of \( \nabla \)-LSTD(\( \lambda \))-learning algorithms. It is shown that \( \nabla \)-LSTD(1) also solves the quadratic program (10).

4) These new algorithms are applicable for models that do not have regeneration. Their asymptotic variance is uniformly bounded over all \( 0 < \beta < 1 \), under general conditions.

Perhaps the most important limitation of the \( \nabla \)-LSTD algorithms is the requirement of partial knowledge of the Markov chain transition dynamics. Section VI contains discussion on how to address this challenge. Fortunately, in many applications, very little knowledge is required (such as in the queuing example discussed in Section V-B). It is also important to note that the algorithms proposed here addresses a more challenging problem of estimating the gradient of the value function.

Finally, a few more remarks about the error rates of these algorithms are in order. From the definition of the value function \( (2) \), it can be expected that \( h_\beta(x) \to \infty \) as \( \beta \to 1 \) for each \( x \in \mathbb{R}^\ell \). This is why approximation methods in reinforcement learning typically take for granted that error will grow at this rate. Moreover, it is observed that variance in reinforcement learning can grow dramatically with the discount factor. In particular, it is shown in [23] and [24] that asymptotic variance in the standard Q-learning algorithm of Watkins is infinite when the discount factor satisfies \( \beta > 1/2 \).

The family of TD(\( \lambda \)) algorithms was introduced in [21] to reduce the variance of earlier methods, but it brings its own potential challenges. Consider [29, Th. 1], which compares the estimate \( h_\beta^\theta \) obtained using TD(\( \lambda \)), with the \( L_2 \)-optimal approximation \( h_\beta^\theta \) obtained using TD(1)

\[
    ||h_\beta^\theta - h_\beta||_\pi \leq \frac{1 - \lambda \beta}{1 - \beta} ||h_\beta^\theta - h_\beta||_\pi. \tag{13}
\]

This bound suggests that the bias can grow as \( (1 - \beta)^{-1} \).

The difficulties are more acute when we come to the average-cost problem. Consider the minimum-norm problem (8) with the relative value function \( h \) in place of \( h_\beta \)

\[
    \theta^* = \arg \min_\theta \|h_\beta^\theta - h\|_\pi^2. \tag{14}
\]

Here, for the TD(\( \lambda \)) algorithm with \( \lambda < 1 \), [6, Th. 3] implies a bound in terms of the “convergence rate” \( \rho \) for the Markov chain

\[
    ||h_\beta^\theta - h||_\pi \leq c(\lambda, \rho)||h_\beta^\theta - h||_\pi \tag{15}
\]

in which \( c(\lambda, \rho) > 1 \) and \( c(\lambda, \rho) \to 1 \) as \( \lambda \to 1 \).

Convergence of TD(1) does not hold for the average cost problem. Specialized algorithms making use of regeneration were introduced in [43, p. 1012], [3], [26].

For any differentiable function function \( f : \mathbb{R}^\ell \to \mathbb{R} \), its gradient is denoted

\[
    \nabla f := \left[ \frac{\partial}{\partial x_1} f, \ldots, \frac{\partial}{\partial x_\ell} f \right]^T. \tag{16}
\]

Under the assumptions imposed in this article, we show that the gradients of the value functions are well behaved: \( \{\nabla h_\beta : 0 < \beta \leq 1\} \) is a bounded collection of functions, and \( \nabla h_\beta \to \nabla h \) uniformly on compact sets. As a consequence, both the bias and variance of the new \( \nabla \)-LSTD(\( \lambda \)) algorithms are bounded over all \( 0 < \beta \leq 1 \).

The remainder of the article is organized as follows: Basic definitions and value function representations are presented in Section II. The \( \nabla \)-LSTD-learning algorithm is introduced in Section III, and the family of \( \nabla \)-LSTD(\( \lambda \)) algorithms are introduced in Section IV. Results from numerical experiments are shown in Section V, and conclusions are contained in Section VI.

**II. REPRESENTATIONS AND APPROXIMATIONS**

We begin with modeling assumptions on the Markov process \( X \), and representations for the value functions \( h_\beta, h \) and their gradients.

**A. Markovian Model and Value Function Gradients**

The evolution equation (1) defines a Markov chain \( X \) with transition semigroup \( \{P^t\} \), where \( P^t(x, A) \) is defined, for all \( t \geq 0 \), any state \( x \in \mathbb{R}^\ell \), and every measurable \( A \subset \mathbb{R}^\ell \), via

\[
    P^t(x, A) := P_x\{X(t) \in A\} := \text{Pr}\{X(t) \in A | X(0) = x\}. \tag{17}
\]

For \( t = 1 \) we write \( P = P^1 \), so that

\[
    P(x, A) = \text{Pr}\{a(x, N(1)) \in A\} \tag{18}
\]

where we recall that \( a : \mathbb{R}^{\ell+m} \to \mathbb{R}^\ell \) defines the dynamics of the Markov chain in (1).

The first set of assumptions ensures that the value functions \( h_\beta \) and \( h \) are well defined. Fix a continuous function \( v : \mathbb{R}^\ell \to [1, \infty) \) that serves as a weighting function. For any measurable function \( f : \mathbb{R}^\ell \to \mathbb{R} \), the \( \nu \)-norm is defined as follows:

\[
    \|f\|_\nu := \frac{\int f(x) \, \nu(x)}{\int \nu(x)} \tag{19}
\]

and the associated Banach space is denoted

\[
    L_\nu^\infty := \{f : \mathbb{R}^\ell \to \mathbb{R} : \|f\|_\nu < \infty\}. \tag{20}
\]
Also, for any measurable function $f$ and measure $\mu$, we write $\mu(f)$ for the integral, $\mu(f) := \int f \, d\mu$.

**Assumption A1:**
The Markov chain $X$ is $v$-uniformly ergodic: It has a unique invariant probability measure $\pi$, and there exists a continuous function $v : \mathbb{R}^\ell \to \mathbb{R}$ and constants $b_0 < \infty$ and $0 < \rho_0 < 1$, such that, for each function $f \in L^\infty$,

$$|\mathbb{E}[f(X(t)) | X(0) = x] - \pi(f)| \leq b_0 \rho_0^t \|f\|_{\infty},$$

for all $x \in \mathbb{R}^\ell$, $t \geq 0$.

Assumption A1 is not a strong assumption; it is essentially equivalent to geometric ergodicity in the usual sense (ignoring a set of measure zero): combine [10, Ths. 15.0.2 and 16.0.1].

There is also an exact equivalence between Assumption A1 and the existence of a “Lyapunov function” [10, Th. 16.0.1]. See [10] and Section V of this article for examples where the assumption holds.

The following consequences are immediate [3], [10].

**Proposition II.1:** Under assumption A1, for any cost function $c$ such that $\|c\|_1 < \infty$, the limit $\gamma$ in (4) exists with $\gamma := \pi(c) < \infty$, and is independent of the initial state $x$. The value functions $h_\beta$ and $h$ exist as expressed in (2) and (5), and they satisfy (3) and (6), respectively.

Moreover, there exists a constant $b_c < \infty$ such that the following bounds hold:

$$|h(x)| \leq b_c v(x),$$

$$|h_\beta(x)| \leq b_c (v(x) + (1 - \beta)^{-1})$$

$$|h_\beta(x) - h_\beta(y)| \leq b_c (v(x) + v(y)), \quad x, y \in \mathbb{R}^\ell.$$

The following operator-theoretic notation will simplify exposition throughout. For any measurable function $f : \mathbb{R}^\ell \to \mathbb{R}$, the new function $P^t f : \mathbb{R}^\ell \to \mathbb{R}$ is defined as the conditional expectation

$$P^t f(x) := \mathbb{E}_x[f(X(t)) | X(0) = x].$$

For any $\beta \in (0,1)$, the resolvent kernel $R_\beta$ is the “$z$-transform” of the semigroup $\{P^t\}$

$$R_\beta := \sum_{t=0}^{\infty} \beta^t P^t.$$

Under the assumptions of Proposition II.1, the discounted-cost value function $h_\beta$ admits the representation

$$h_\beta = R_\beta c,$$

and similarly, for the relative value function $h$ we have

$$h = R[c - \eta].$$

where we write $R \equiv R_1$ when $\beta = 1$ [3], [10], [30].

The representations (18) and (19) are valuable in deriving the LSTD-learning algorithms [2], [3], [31]. Analogous representations for the gradients are obtained in this article

$$\nabla h_\beta = \nabla[R_\beta c], \quad \nabla h = \nabla[Re].$$

**B. Representation for the Gradient of a Value Function**

In this section, we describe the construction of operators $\Omega$ and $\Omega_\beta$, which satisfy the following:

$$\nabla h_\beta = \nabla[R_\beta c] = \Omega_\beta \nabla c, \quad \nabla h = \nabla[Re] = \Omega \nabla c. \quad (20)$$

A more detailed account is given in Section III-B, and a complete exposition of the underlying theory together with the formal justification of the existence and the relevant properties of $\Omega$ and $\Omega_\beta$ can be found in [28].

For the sake of simplicity, here we restrict our discussion to $h_\beta$ and its gradient. But it is not hard to see that the construction below easily generalizes to $\beta = 1$; again (see Section III-B and [28] for the relevant details).

We require the following further assumptions.

**Assumption A2:**

A2.1: The noise process $N$ is independent of $X(0)$.

A2.2: The function $a$ is continuously differentiable in its first variable, with

$$\sup_{x,n} \|\nabla_x a(x, n)\| < \infty$$

where $\| \cdot \|$ is any matrix norm, and the $\ell \times \ell$ matrix $\nabla_x a$ is defined as

$$[\nabla a_x(x, n)]_{i,j} := \partial a_x(x, n)_{i,j}, \quad 1 \leq i, j \leq \ell. \quad (21)$$

The first assumption A2.1, is critical so that the initial state $X(0) = x$ can be regarded as a variable, with $X(t)$ a continuous function of $x$. This together with A2.2 allows us to define the sensitivity process $\{S(t)\}$, where, for each $t \geq 0$

$$S_{i,j}(t) := \partial X_i(t) / \partial X_j(0), \quad 1 \leq i, j \leq \ell. \quad (22)$$

The evolution equations (1) imply that the sensitivity process evolves as a random linear system

$$S(t + 1) = A(t + 1) S(t), \quad t \geq 0$$

with initial condition $S(0) = I$, where the $\ell \times \ell$ matrix $A(t)$ is defined as in assumption A2.2, by

$$A(t) := \nabla_a(X(t - 1), N(t)).$$

For any $C^1$ function $f : \mathbb{R}^\ell \to \mathbb{R}$, denote

$$\nabla f(X(t)) := S(T(t)) \nabla f(X(t)). \quad (23)$$

It follows from the chain rule that this coincides with the gradient of $f(X(t))$ with respect to the initial condition $x$

$$[\nabla f(X(t))]_{i,j} = \partial f(X(t)) / \partial X_i(0), \quad 1 \leq i, j \leq \ell. \quad (24)$$

Equation (24) motivates the introduction of a semigroup $\{Q^t : t \geq 0\}$ of operators, whose domain includes functions $g : \mathbb{R}^\ell \to \mathbb{R}^\ell$ of the form $g = [g_1, \ldots, g_\ell]^T$, with $g_i \in L_{\infty}$ for each $i$. For $t = 0, Q^0$ is the identity operator, and for $t \geq 1$

$$Q^t g(x) := \mathbb{E}_x[S^T(t)g(X(t))]. \quad (26)$$
Provided we can exchange the gradient and the expectation, following (25), we have: 
\[
\frac{\partial}{\partial x_i} E_x[f(X(t))] = E_x[\nabla \delta f(X(t))]_i, \quad 1 \leq i \leq \ell
\]
and consequently, the following elegant formula is obtained: 
\[
\nabla P^t f(x) = E_x[\nabla \delta f(X(t))] = Q^t \nabla f(x), \quad x \in \mathbb{R}^\ell. \quad (27)
\]
Justification requires minimal assumptions on the function $f$. The proof of Proposition II.2 is based on Lemmas A.1 and A.2 contained in Appendix A.

Proposition II.2: Suppose that Assumptions A1 and A2 hold, and that $f^2$ and $\|\nabla f\|_2^2$ both lie in $L^\infty$. Then (27) holds, and $\nabla P^t f(x)$ is continuous as a function of $x \in \mathbb{R}^\ell$.

Proof: The proof uses Lemma A.2 in the Appendix, and a variant of the truncation argument of [28]. Let $\{\chi_n : n \geq 1\}$ be a sequence of functions satisfying, for each $n$.

1) $\chi_n$ is a continuous approximation to the indicator function on the set $R_n$, where

\[
R_n := \{x \in \mathbb{R}^\ell : |x_i| \leq n, \quad 1 \leq i \leq \ell\}
\]
in the sense that $0 \leq \chi_n(x) \leq 1$ for all $x$, $\chi_n(x) = 1$ when $x \in R_n$, and $\chi_n(x) = 0$ when $x \in R_n^c$.

2) $\nabla \chi_n$ is continuous and uniformly bounded:

\[
sup_{x \in \mathbb{R}^\ell} \|\nabla \chi_n(x)\| < \infty.
\]

On denoting $f_n = \chi_n f$, we have

\[
\nabla f_n = \chi_n \nabla f + f \nabla \chi_n
\]
which is bounded and continuous under the assumptions of the proposition. An application of the mean value theorem combined with dominated convergence allows us to exchange differentiation and expectation

\[
\frac{\partial}{\partial x_i} E_x[f_n(X(t))] = E_x\left[\frac{\partial}{\partial x_i} f_n(X(t))\right], \quad 1 \leq i \leq \ell.
\]
This identity is equivalent to (27) for $f_n$.

Under the assumptions of the proposition, there is a constant $b$ such that $\|\nabla f_n(x)\|^2 \leq b v(x)$ for each $n$ and $x \in \mathbb{R}^\ell$. Applying the dominated convergence theorem once more gives

\[
Q^t \nabla f(x) = \lim_{n \to \infty} Q^t \nabla f_n(x), \quad x \in \mathbb{R}^d.
\]
The limit is continuous by Lemma A.2, and an application of [28, Lemma 3.6] completes the proof.\[ \]

Proposition II.2 (a) strongly suggests the representation $\nabla h_\beta = \Omega_\beta \nabla c$ in (20) holds, with

\[
\Omega_\beta := \sum_{t=0}^\infty \beta^t Q^t.
\]
This is indeed justified (under additional assumptions) in [28, Th. 2.4], and it forms the basis of the $\nabla$-LSTD-learning algorithms developed in this article.

Similarly, the representation $\nabla h = \Omega \nabla c$ with $\Omega = \Omega_1$ for the gradient of the relative value function $h$ is derived, under appropriate conditions, in [28, Th. 2.3].

### III. DIFFERENTIAL LSTD-LEARNING

In this section, we develop the new differential LSTD (or $\nabla$-LSTD, or “grad-LSTD”) learning algorithms for approximating the value functions $h_\beta$ and $h$, cf. (2), (5). The algorithms are presented first, with supporting theory in Section III-B. We concentrate mainly on the family of discounted-cost value functions $h_\beta, 0 < \beta < 1$. The extension to the case of the relative value function $h$ is briefly discussed in Section III-C.

#### A. Differential LSTD Algorithms

We begin with a review of the standard LSTD-learning algorithm, cf. [2], [3]. We assume that the following are given. A target number of iterations $T$ together with $T$ samples from the process $X$, the discount factor $\beta$, the functions $\psi$, and a gain sequence $\{\alpha_t\}$. Throughout the article, the gain sequence $\{\alpha_t\}$ is taken to be $\alpha_t = 1/t, t \geq 1$.

Algorithm 1 is equivalent to the LSTD(1) algorithm of [32]; see Section IV and [23], [25] for more details.

To simplify discussion we restrict to a stationary setting for the convergence results in this article.

Proposition III.1: Suppose that assumption A1 holds, and that the functions $c^2$ and $\|\psi\|_2^2$ are in $L^\infty$. Suppose moreover that the matrix $M = E[\psi(X)^T \psi(X)]$, $X \sim \pi$, is of full rank.

Then, there exists a version of the pair process $(X, \varphi) = \{(X(t), \varphi(t))\}$ that is stationary on the two-sided time axis, and for any initial choice of $b(0), \varphi(0) \in \mathbb{R}^d$ and $M(0)$ positive definite, Algorithm 1 is consistent

\[
\varphi^* = \lim_{t \to \infty} M^{-1}(t)b(t) \quad \text{a.s.}
\]
where $\varphi^*$ is the least squares minimizer in (8).

Proof: The existence of a stationary solution $X$ on the two-sided time interval follows directly from $\nu$-uniform ergodicity, and we then define, for each $t \geq 0$

\[
\varphi(t) = \sum_{i=0}^t \beta^i \psi(X(t-i)).
\]
The optimal parameter can be expressed $\varphi^* = M^{-1}b$ in which $b = E[\varphi(t)\psi(X(t))]$, where the expectation is in steady state, so the result follows from the law of large numbers for this stationary ergodic process.

In the construction of the LSTD algorithm, the optimization problem (8) is cast as a minimum-norm problem in the Hilbert
Algorithm 2: $\nabla$-LSTD Algorithm.

Input: Initial $b(0) \in \mathbb{R}^d$, $\varphi(0) \in \mathbb{R}^{d \times d}$, $M(0) d \times d$ positive definite, and $t = 1$

1: repeat
2: $\varphi(t) = \beta A(t) \varphi(t - 1) + \nabla \psi(X(t));$
3: $b(t) = b(t - 1) + \alpha_t (\varphi^T(t) \nabla c(X(t)) - b(t - 1));$
4: $M(t) = M(t - 1) + \alpha_t (\nabla \psi(X(t))^T \nabla \psi(X(t)) - M(t - 1));$
5: $t = t + 1$
6: until $t \geq T$

Output: $\theta = M^{-1}(T)b(T)$

space

$L_{\pi}^2 = \{g: \mathbb{R}^\ell \to \mathbb{R} : \|g\|_{\pi}^2 = \langle g, g \rangle_\pi < \infty\}$

with inner-product, $(f, g)_\pi := \int f(x)g(x)\pi(dx)$.

The $\nabla$-LSTD algorithm presented next is based on a minimum-norm problem in a different Hilbert space. For $C^1$ functions $f$, $g$, with each $[\nabla f]_i$, $[\nabla g]_i \in L_{\pi}^2$, $1 \leq i \leq \ell$, define the inner product

$\langle f, g \rangle_{\pi, 1} = \int \nabla f(x)^T \nabla g(x)\pi(dx)$

with the associated norm $\|f\|_{\pi, 1} := \sqrt{\langle f, f \rangle_{\pi, 1}}$. We let $L_{\pi, 1}^2$ denote the set of functions with finite norm

$L_{\pi, 1}^2 = \{h : \mathbb{R}^\ell \to \mathbb{R} : \|h\|_{\pi, 1}^2 < \infty\}.$

(29)

Two functions $f, g \in L_{\pi, 1}^2$ are considered identical if $\|f - g\|_{\pi, 1} = 0$. In particular, this is true if the difference $f - g$ is a constant independent of $x$.

The “differential” version of the least-squares problem in (8), given as the nonlinear program (10), can now be recast as

$\theta^* = \arg \min_{\theta} \|h_0^\theta - h_\beta\|_{\pi, 1}.$

(30)

Given a target number of iterations $T$ together with $T$ samples from the process $X$, the discount factor $\beta$, the functions $\psi$, and a gain sequence $\{\alpha_t\}$, the $\nabla$-LSTD algorithm, defined in Algorithm 2, solves (30), with

$[\nabla \psi(x)]_{i,j} := \frac{\partial}{\partial x_i} \psi_j(x), \quad x \in \mathbb{R}^\ell.$

(31)

Once the estimate of $\theta^*$ is obtained from Algorithm 2, the required estimate of $h_\beta$ is obtained as $h_0^\theta = \theta^* \psi + \kappa(\theta)$, where

$\kappa(\theta) = -\pi (h_0^\theta) + \eta/(1 - \beta)$

(32)

with $\eta = \pi(c)$ as in (4), and with the two means $\eta$ and $\pi(h_0^\theta)$ given by the results of the following recursive estimates:

$T_{\beta}(t) = T_{\beta}(t - 1) + \alpha_t \left(h_0^{\theta(t)}(X(t)) - T_{\beta}(t - 1)\right)$

(33)

$\eta(t) = \eta(t - 1) + \alpha_t (c(X(t)) - \eta(t - 1)).$

(34)

It is immediate that $\eta(t) \to \eta$, a.s., as $t \to \infty$, by the law of large numbers for $\nu$-uniformly ergodic Markov chains [10].
and also implies that this gradient is continuous as a function of $x$. Assumption A3.2 implies that the right-hand side converges to $\Omega_\beta \nabla c(x)$ as $n \to \infty$. The function $\Omega_\beta \nabla c$ is continuous in $x$, since the limit is uniform on compact subsets of $\mathbb{R}^\ell$ (recall that $\nu$ is continuous). Lemma 3.6 of [28] then completes the proof.

A stationary realization of the algorithm is established next. Lemma III.4 follows immediately from the assumptions. The nonrecursive expression for $\varphi(t)$ in (41) is immediate from the recursions in Algorithm 2.

Lemma III.4: Suppose that assumptions A1–A3 hold, and that $\|\psi\|_2^2$ and $\|
abla \psi\|_2^2$ are in $L^\infty_{\omega, \mu}$. Then there is a version of the pair process $(X, \varphi)$ that is stationary on the two-sided time line, and for each $t \in \mathbb{Z}$

$$\varphi(t) = \sum_{k=0}^{\infty} \beta^k [\Theta^{t-k} S(k)] \nabla \psi(X(t-k)) \tag{41}$$

where $\Theta^{t-k} S(k) = A(t) A(t-1) \cdots A(t-k+1)$.

The remainder of this section consists of a proof of the following proposition which establishes the convergence of the $\nabla$-LSTD algorithm.

Proposition III.5: Suppose that assumptions A1–A3 hold, and that $c^2, \|\nabla c\|_2^2, \|\psi\|_2^2$ and $\|
abla \psi\|_2^2$ are in $L^\infty_{\omega, \mu}$. Suppose moreover that the matrix $M$ in (35) is of full rank. Then, for the stationary process $(X, \varphi)$, the $\nabla$-LSTD-learning algorithm is consistent: For any initial $b(0) \in \mathbb{R}^d$ and $M(0) > 0$

$$\theta^* = \lim_{t \to \infty} M^{-1}(t) b(t) \text{ a.s.}$$

where $\theta^*$ is the least squares minimizer in (10). Moreover, with probability one

$$\eta = \lim_{t \to \infty} \eta(t), \quad \pi(h^0) = \lim_{t \to \infty} \tau_\beta(t)$$

and hence $\lim_{t \to \infty} \{-\tau_\beta(t) + \eta(t)/(1 - \beta)\} = \kappa(\theta^*)$.

We begin by obtaining alternative representations for $b$ defined in (38). The proof of the following Lemma III.6 is contained in Appendix B.

Lemma III.6: Under the assumptions of Proposition III.5

$$b^T = \sum_{t=0}^{\infty} \beta^t \mathbb{E}[\{S(t) \nabla c(X(t))\}^T \nabla \psi(X(0))]$$

$$= \mathbb{E}[\nabla c(X(0))]^T \varphi(0). \tag{42}$$

Proof of Proposition III.5. Lemma III.6 combined with the stationarity assumption implies that

$$\lim_{T \to \infty} \frac{1}{T} b(t) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \varphi(t) \nabla c(X(t))$$

$$= \mathbb{E}[\varphi(0) \nabla c(X(0))].$$

Similarly, for each $T \geq 1$ we have

$$M(T) = M(0) + \sum_{t=1}^{T} (\nabla \psi(X(t)))^T \nabla \psi(X(t))$$

and by the law of large numbers we once again obtain

$$\lim_{T \to \infty} \frac{1}{T} M(T) = M.$$

Combining these results establishes $\theta^* = \lim_{t \to \infty} M^{-1}(t) b(t)$.

Convergence of $\{\eta(t)\}$ in (34) is identical, and convergence of $\{\tau_\beta(t)\}$ in (33) also follows from the law of large numbers since we have convergence of $\theta^*(t)$.

C. Extension to Average Cost

The $\nabla$-LSTD recursion of Algorithm 2 is also consistent in the case $\beta = 1$, which corresponds to the relative value function $h$ in place of the discounted-cost value function $h_\beta$. Although we do not repeat the details of the analysis here, we observe that nowhere in the proof of Proposition III.5 do we use the assumption that $\beta < 1$. Indeed, it is not difficult to establish that, under the conditions of the proposition, the $\nabla$-LSTD-learning algorithm is also convergent with $\beta = 1$, and that the limit solves the quadratic program

$$\theta^* = \arg \min_{\theta} \|h^0 - h\|_{\pi, 1}.$$
which achieves

\[
\mathbb{E} \left[ \zeta^T(t) (\nabla c(X(t)) + \beta A^T(t + 1) \nabla h^\theta_{\beta} (X(t + 1))) - \nabla h^\theta_{\beta} (X(t))) \right] = 0 \tag{44}
\]

will be estimated, where the above expectation is again in steady state. By its construction, \( \zeta \) will be adapted to \( X \). We call \( \zeta \) the sequence of **eligibility matrices**, borrowing language from the standard LSTD(\( \lambda \))-learning literature [2], [21], [40].

Motivation for the minimum-norm criterion (30) is clear, but algorithms that solve this problem often suffer from high variance. The Galerkin approach is used because it is simple, generally applicable, and it is observed that the variance of the algorithm is often significantly reduced with \( \lambda < 1 \).

It is important to note, as we also discuss below, that the process \( \zeta \) will depend on the value of \( \lambda \), so the LSTD(\( \lambda \)) [respectively, \( \nabla \)-LSTD(\( \lambda \))] algorithms with different \( \lambda \) will converge to different parameter values \( \theta^* = \theta^*(\lambda) \), satisfying the corresponding versions of (44).

### A. Differential LSTD(\( \lambda \)) Algorithms

Recall the standard algorithm introduced in [32]; see also [2], [3]. Given a target number of iterations \( T \) together with \( T \) samples from the process \( X \), the discount factor \( \beta \), the functions \( \psi \), a gain sequence \( \{ \alpha_i \} \), and \( \lambda \in [0, 1] \).

The asymptotic consistency of Algorithm 3 is established, e.g., in [32] and [41]. Note that, unlike in Algorithms 1 and 2, here there is no guarantee that \( M(t) \) is positive definite for all \( t \), so by the output value of \( \theta = M^{-1}(T)b(T) \) we mean that obtained by using the pseudoinverse of \( M(T) \); and similarly for Algorithm 4 presented next.

The differential analog of Algorithm 3 is very similar; recall the definition of \( \nabla \psi(x) \) in (31).

As with Algorithm 2, after obtaining the estimate of \( \theta \) from Algorithm 4, the required estimate of \( h_{\beta} \) is formed based on the recursions in (32)–(34).

### B. Derivation and Analysis

For any \( \lambda \in [0, 1] \), the parameter vector \( \theta^* = \theta^*(\lambda) \) that solves (44) is a Galerkin approximation to the exact solution which solves the fixed point equation (43).

The proof of the first part of Proposition IV.1 below follows from the assumptions. In particular, the nonrecursive expression for \( \zeta(t) \) is a consequence of the recursions in Algorithm 4. The proof of the second part of the proposition follows from (44).

**Proposition IV.1:** Suppose that assumptions A1–A3 hold, and that \( \| \psi \|^2 \) and \( \| \nabla \psi \|^2 \) are in \( L^2 \). Then

(i) There is a stationary version of the pair process \( (X, \zeta) \) on the two-sided time axis, and for each \( t \in \mathbb{Z} \) we have

\[
\zeta(t) = \sum_{k=0}^{\infty} (\beta \lambda)^k \left[ \Theta^{t-k} S(k) \right] \nabla \psi(X(t-k))
\]

where \( \Theta^{t-k} S(k) = A(t)A(t-1) \cdots A(t-k+1) \).

(ii) The optimal parameter vector \( \theta^* \) that satisfies (44) is any solution to \( M\theta^* = b \), in which

\[
M = \mathbb{E}[(\nabla \psi(X(t)) - \beta A^T(t+1)\nabla \psi(X(t+1)))^T \zeta(t)]
\]

and

\[
b = \mathbb{E}[(\zeta(t))^T \nabla c(X(t))] \tag{46}
\]

where the expectations are under stationarity.

The following then follows from the law of large numbers.

**Proposition IV.2:** Suppose that the assumptions of Proposition III.5 hold. Suppose moreover that the matrix \( M \) appearing in (45) is of full rank. Then, for each initial conditions \( b(0) \in \mathbb{R}^d \) and \( M(0) \in \mathbb{R}^{d \times d} \), the \( \nabla \)-LSTD(\( \lambda \)) Algorithm 4 is consistent

\[
\lim_{t \to \infty} M^{-1}(t) b(t) = \theta^* \quad \text{a.s.}
\]

where \( \theta^* = \theta^*(\lambda) \) solves (44).

This limit holds both for the stationary version \( (X, \varphi) \) defined in Proposition IV.1, and also for \( \varphi \)-almost all initial \( (X(0), \zeta(0)) \), where \( \varphi \) denotes the marginal for the stationary version \( (X, \varphi) \).

### C. Optimality of \( \nabla \)-LSTD(1)

Although different values of \( \lambda \) in LSTD(\( \lambda \)) lead to different parameter estimates \( \theta^* = \theta^*(\lambda) \), it is known that in the case \( \lambda = 1 \) the parameter estimates obtained using the standard LSTD(\( \lambda \)) algorithm converge to the solution of the minimum-norm problem (8), cf. [23], [29]. Similarly, it is shown here that the parameter estimates obtained using the \( \nabla \)-LSTD(1) algorithm converge to the solution of the minimum-norm problem (30).
 Suppose that the assumptions of Proposition III.5 hold. Then, the sequence of parameters \( \theta = \{ \theta(t) \} \) obtained using the \( \nabla\text{-LSTD}(\lambda) \) Algorithm 4, converges to the solution of the minimum-norm problem (30).

Proof: From Proposition IV.2, the estimates \( \hat{\theta} \) obtained using the \( \nabla\text{-LSTD}(\lambda) \) algorithm converge to \( \theta^* = M^{-1}b \), where \( M \) and \( b \) are defined in (45) and (46), and \( \zeta(t) \) defined by the recursion in Algorithm 4. It remains to be shown that this coincides with the parameter that solves (30) in the case \( \lambda = 1 \).

Substituting the identity
\[
\zeta(t + 1) = \beta A(t + 1) \zeta(t) + \nabla \psi(X(t + 1))
\]
(47)
in (45), gives the following representation:
\[
M = -\beta E[(A^T(t + 1)\nabla \psi(X(t + 1)))^T \zeta(t)] + E[(\nabla \psi(X(t)))^T \zeta(t)]
\]
\[
= -\beta E[(A^T(t + 1)\nabla \psi(X(t + 1)))^T \zeta(t)] + \beta E[(A^T(t)\nabla \psi(X(t)))^T \zeta(t - 1)] + E[(\nabla \psi(X(t)))^T \nabla \psi(X(t))]
\]
where the last equality is obtained using time stationarity of \( X \). Therefore, the matrix \( M \) obtained using the \( \nabla\text{-LSTD}(1) \) algorithm coincides with the matrix \( M \) of \( \nabla\text{-LSTD} \).

To obtain the required representation for \( b \), recall that \( \zeta(t) = \varphi(t) \), where the former is defined in (47) and the latter in the recursion of Algorithm 2. Applying Lemma III.6, it follows that the vector \( b \) of the \( \nabla\text{-LSTD}(1) \) algorithm (46) coincides with the vector \( b \) of \( \nabla\text{-LSTD} \) algorithm (42).

**V. NUMERICAL RESULTS**

Collected together here are results from several numerical experiments, which illustrate the general theory of the previous sections and also suggest possible extensions.

Since, under general conditions, all estimates considered obey a central limit theorem [42], we use the asymptotic variance to be the primary figure of merit in evaluating performance. The relevant variances are estimated by collecting data from many independent runs of each algorithm.

Specifically, we show comparisons between the performance achieved by LSTD, \( \nabla\text{-LSTD}, \text{LSTD}(\lambda), \) and the \( \nabla\text{-LSTD}(\lambda) \) algorithms. In examples where there is regeneration (the Markov chain \( X \) visits some state infinitely often), we replace the LSTD algorithm with the regenerative LSTD algorithm of [3], [26]. The regenerative algorithm is found to have reduced variance in these experiments.

The standard TD(\( \lambda \)) algorithm was also considered, but in all examples its variance was found to be several orders of magnitude greater than alternatives. The matrix-gain variant with minimal asymptotic variance is precisely LSTD(\( \lambda \)) [23], [32]. This was found to have better performance, and was therefore used for comparisons; the reader is referred to [23, Sec. 2.4] for details on the relationship between TD(\( \lambda \)) and LSTD(\( \lambda \)) algorithms, and their asymptotic variances.

We also consider two extensions of \( \nabla\text{-LSTD} \) for a specific example: The approximation of the relative value function for the speed-scaling model of [44]. First, for this reflected process evolving on \( \mathbb{R}_+ \), it is shown that the sensitivity process \( S \) can be defined, subject to conditions on the dynamics near the boundary. Second, the algorithm is tested in a discrete state space setting. There is no apparent justification for this approach, but it performs remarkably well in simulations.

### A. Linear Model

A scalar linear model offers perhaps the clearest illustration of the performance of the \( \nabla\text{-LSTD} \)-learning algorithm, demonstrating its superior convergence rate compared to the standard LSTD algorithm.

Consider the scalar linear process
\[
X(t + 1) = aX(t) + N(t + 1), \quad t \geq 1
\]
where \( a \in (0, 1) \) is a constant and \( N(t) \) is i.i.d. \( \mathcal{N}(0, 1) \). The cost function is taken to be quadratic, \( c(x) = x^2 \), and for the basis of the approximating function class is chosen as \( \psi(x) = (1, x^2)^T \). The true value function \( h_\beta \) turns out to also be quadratic and symmetric, which means that it can be expressed exactly in terms of \( \psi \), as \( h_\beta = h^\beta_\varphi \), with
\[
h^\beta_\varphi(x) = \sum_j \theta_j \varphi_j(x) = \theta_1^* + \theta_2^* x^2
\]
for appropriate \( \varphi^* \in \mathbb{R}^2 \); cf. (7). The constant term, \( \theta_1^* \), can be estimated as \( \kappa(\theta) \) using (32) in the \( \nabla\text{-LSTD} \) algorithm. Therefore, the interesting part of the problem is to estimate the optimal value of the second parameter \( \theta_2^* \).

For this linear model, line 2 of LSTD (Algorithm 1) and \( \nabla\text{-LSTD} \) (Algorithm 2) become
\[
\varphi(t) = \beta \varphi(t - 1) + \psi(X(t)) \quad \text{for LSTD}
\]
\[
\varphi(t) = \beta a \varphi(t - 1) + \nabla \psi(X(t)) \quad \text{for } \nabla\text{-LSTD}.
\]
Although both of these algorithms are consistent, there are two differences which immediately suggest that the asymptotic variance of \( \nabla\text{-LSTD} \) should be much smaller than that of LSTD. First, the additional discounting factor \( a \) appearing in (49), but absent in (48), is the reason why the asymptotic variance of the \( \nabla\text{-LSTD} \) is bounded over \( 0 < \beta < 1 \), whereas that of the standard LSTD grows without bound as \( \beta \to 1 \). Second, the gradient reduces the growth rate of each function of \( x \); in this case, reducing the quadratic growth of \( c \) and \( \psi \) to the linear growth of their derivatives.

In the numerical experiments surveyed here, we use \( a = 0.7 \), and two different discounting factors: \( \beta = 0.9 \) and \( \beta = 0.99 \). The optimal parameters can be computed explicitly, giving \( \theta^* = (16.1, 1.79)^T \) when \( \beta = 0.9 \), and \( \theta^* = (192.27, 1.9421)^T \) when \( \beta = 0.99 \). The histogram of the estimated value of \( \theta_2 \) was computed based on 1000 repetitions of the same experiment, where the output of each algorithm was recorded after \( T = 10^3 \) and after \( T = 10^6 \) iterations. The results are shown in Fig. 1.
With \( \beta = 0.9 \) it was found that the variance of \( \theta_2 \) using the standard LSTD algorithm is about ten times the variance using the \( \nabla \)-LSTD algorithm. Consequently, \( \nabla \)-LSTD-learning is about ten times faster than LSTD in this example. This difference in performance grows with larger \( \beta \), as observed on the two histograms on the right-hand side of Fig. 1.

In conclusion, in contrast to the standard LSTD algorithm, the asymptotic variance of \( \nabla \)-LSTD in this example is bounded uniformly over \( 0 < \beta < 1 \), and the algorithm can also be used to estimate the relative value function (6).

We next consider an example with nonlinear dynamics.

### B. Dynamic Speed Scaling

Dynamic speed scaling refers to control techniques for power management in computer systems. The goal is to control the processing speed so as to optimally balance energy and delay costs; this can be done by reducing (or increasing) the processor speed at times when the workload is small (respectively, large).

For our purposes, speed scaling is a simple stochastic control problem—namely, a single-server queue with controllable service rate.

This example was considered in [44] with the goal of minimizing the average cost (4). Approximate policy iteration was used to obtain the optimal control policy, and a regenerative form of the LSTD-learning was used to provide an approximate relative value function \( h \) at each iteration.

The underlying discrete-time Markov decision process model is as follows: At each time \( t \), the state \( X(t) \) is the (not necessarily integer valued) queue length, which can also be interpreted more generally as the size of the workload in the system; \( N(t) \geq 0 \) is number of job arrivals; and \( U(t) \) is the service completion at time \( t \), which is subject to the constraint \( 0 \leq U(t) \leq X(t) \). The evolution equation is the controlled random walk

\[
X(t+1) = X(t) - U(t) + N(t+1), \quad t \geq 0. \tag{50}
\]

Under the assumption that \( N \) is i.i.d. and that \( U = \{U(t)\} \) is obtained using a state feedback policy, \( U(t) = f(X(t)) \), the controlled model is a Markov chain of the form (1).

In the experiments that follow in Sections V-B1 and V-B2, we consider the problem of approximating the relative value function \( h \), for a fixed state feedback policy \( f \), so \( \beta = 1 \) throughout.

We consider the cost function \( c(x, u) = x + u^2/2 \), and feedback law \( f \) given by

\[
f(x) = \min\{x, 1 + \varepsilon \sqrt{x}\}, \quad x \in \mathbb{R} \tag{51}
\]

with \( \varepsilon > 0 \). This is similar in form to the optimal average-cost policy computed in [44], where it was shown that the value function is well approximated by the function \( h^\theta(x) = \theta^T \psi(x) \) for some \( \theta \in \mathbb{R}^2_+ \), and \( \psi(x) = (x^{1/2}, x^T) \). As in the linear example, the gradient \( \nabla \psi(x) = (\frac{1}{2}x^{1/2}, 1)^T \) has slower growth as a function of \( x \).

On a more technical note, we observe that implementation of the \( \nabla \)-LSTD algorithms requires attention to the boundary of the state space. The sensitivity process \( S \) defined in (21) requires that the state space be open, and that the dynamics are smooth. Both of these assumptions are violated in this example. However, with \( X(0) = x \), we do have a representation for the right derivative, \( S(t) := \partial_x^+ X(t)/\partial x \), which evolves according to the recursive equation

\[
S(t+1) = A(t+1)S(t) = \left[ 1 - \frac{d^+ f(X(t))}{\partial x} \right] S(t) \tag{52}
\]

where the “+” again denotes right derivative. Therefore, we adopt the convention

\[
A(t+1) = 1 - \frac{d^+ f(X(t))}{\partial x}. \tag{53}
\]

We begin with the case in which the marginal of \( N \) is exponential. In this case, the right derivatives and ordinary derivatives coincide a.e. The regenerative LSTD algorithm used in [44] is not applicable in this case because there is no state that is visited infinitely often with probability one. We therefore restrict our comparisons to the LSTD(\( \lambda \)) algorithms.

1) Exponential Arrivals: Suppose the \( N(t) \) are i.i.d. Exponential(1) random variables, and that \( X \) evolves on \( \mathbb{R}_+ \) according to (50) and (51). The derivatives \( A(t) \) in (53) become

\[
A(t+1) = 1\{X(t) > \tilde{\varepsilon}\} \left[1 - \frac{1}{2} \varepsilon X(t)^{-1/2}\right] \tag{54}
\]

where \( \tilde{\varepsilon} = \frac{1}{2}(\varepsilon + \sqrt{\varepsilon^2 + 4}) \) and \( 1 \) is the indicator function.

For the implementation of the \( \nabla \)-LSTD Algorithm 2, we note that the recursion for \( \varphi \)

\[
\varphi(t+1) = A(t+1)\varphi(t) + \nabla \psi(X(t+1)) \tag{55}
\]

regenerates: Based on (54), \( \varphi(t+1) = \nabla \psi(X(t+1)) \) when \( X(t) \leq \tilde{\varepsilon} \). The second recursion in Algorithm 2 becomes

\[
b(t+1) = b(t) + \alpha t+1 (b(t) - \nabla c(X(t+1))\varphi(t+1))
\]
Algorithm 5: LSTD(λ) Algorithm for Average Cost.

**Input:** Initial η(0) ∈ ℜ+, b(0), φ(0), ηφ(0) ∈ ℜd, M(0) d × d positive definite, and t = 1

1: repeat
2: η(t) = (1 − αc)η(t − 1) + αcc(X(t))
3: ηφ(t) = (1 − αc)ηφ(t − 1) + αcψ(X(t))
4: \( \tilde{\psi}(t) := \tilde{\psi}(X(t)) - \eta(t) \)
5: \( \zeta(t) = \lambda \zeta(t - 1) + \psi(X(t)) \)
6: \( b(t) = (1 - \alpha) b(t - 1) + \alpha \zeta(t) c(X(t)) - \eta(t) \)
7: \( M(t) = (1 - \alpha) M(t - 1) + \alpha \zeta(t) [\tilde{\psi}(X(t)) - \psi(X(t + 1))]^T \)
8: \( t = t + 1 \)
9: until \( t \geq T \)

**Output:** \( \theta = M^{-1}(T)b(T) \)

Fig. 2. Histograms of the parameter estimates obtained using the LSTD and \( \nabla \cdot \text{LSTD} \) algorithms after \( T = 10^5 \) iterations, under the stationary policy \( \{51\} \) with \( \varepsilon = 0.5 \); \( N \) is i.i.d. exponential.

in which
\[
\nabla c(X(t)) = 1 + f(X(t)) \nabla f(X(t))
\]
\[
\nabla f(X(t)) = 1 \{X(t) \leq f(X(t))\} + \frac{\varepsilon}{2} \frac{1}{\sqrt{X(t)}} 1\{X(t) > \varepsilon\}.
\]

(56)

Implementation of the \( \nabla \cdot \text{LSTD}(\lambda) \) Algorithm 4 uses similar modifications, with \( \{A(t)\} \) and \( \{\nabla c(X(t))\} \) obtained using (54) and (56).

Various forms of the TD(λ) algorithms with \( \lambda \in [0, 1] \) were implemented for comparison, but as reasoned in Section I, all of them appeared to have infinite asymptotic variance. Implementation of the LSTD(λ) algorithm resulted in improved performance. Since this is an average-cost problem, Algorithm 3 must be modified slightly [3], [6], [32].

Other than taking \( \beta = 1 \), the main difference between Algorithms 3 and 5 is that we have replaced the cost function \( c(X(t)) \) with its centered version, \( c(X(t)) - \eta(t) \), where \( \eta(t) \) is the estimate of the average cost after \( t \) iterations. While this is standard for average cost problems, we have similarly replaced the basis function \( \psi \) with \( \tilde{\psi} \) to restrict the growth rate of the eligibility vector \( \zeta(t) \), which in turn reduces the variance of the estimates \( \theta = \{\theta(t)\} \). This is justified because the approximate value functions \( h_{\theta} \) differ from \( h_{\phi} = \theta^T \psi \) only by a constant term, and the relative value function is unique only up to additive constants. Experiments where \( \psi \) was used instead of \( \tilde{\psi} \) resulted in worse performance.

Fig. 2 shows the histogram of the estimates for \( \theta_1 \) and \( \theta_2 \) obtained using \( \nabla \cdot \text{LSTD}-\text{learning}, \text{LSTD}(0)-\text{learning}, \) and \( \nabla \cdot \text{LSTD}(\lambda)-\text{learning}, \lambda = 0 \) and 0.5, after \( T = 10^5 \) time steps.

As noted earlier in Section IV-C, we observe that, as expected, different values of \( \lambda \) lead to different parameter estimates \( \theta^*(\lambda) \), for both the LSTD(λ) and the \( \nabla \cdot \text{LSTD}(\lambda) \) classes of algorithms.

**2) Geometric Arrivals:** In [44], the authors consider a discrete state space, with \( N(t) \) geometrically distributed on an integer lattice \( \{0, \Delta, 2\Delta, \ldots\} \), \( \Delta > 0 \). In this case, the theory developed for the \( \nabla \cdot \text{LSTD} \) algorithm does not fit the model since we have no convenient representation of a sensitivity process. Nevertheless, the algorithm can be run by replacing gradients with ratios of differences. In particular, in implementing the algorithm we substitute the definition (54) with
\[
A(t) = 1 - \frac{[f(X(t) + \Delta) - f(X(t))] / \Delta}{\nabla c}
\]
and \( \nabla c \) is approximated similarly. For the distribution of \( N(t) \), we take, \( P(N(t) = n\Delta) = (1 - p_A)^{n-1}p_A; \) the values \( p_A = 0.04 \) and \( \Delta = 1/24 \) were chosen, so that \( E[N(t)] = 1 \).

The sequence of steps followed in the regenerative LSTD-learning algorithm are similar to Algorithm I [3], [44].

The eligibility vector \( \varphi(t) \) regenerates (i.e., resets to 0) every time the queue empties. The regenerative LSTD(λ) algorithm is obtained by making similar modifications—namely, replacing Line 5 of Algorithm 5 with
\[
\zeta(t) = 1 \{X(t - 1) \neq 0\} \lambda \zeta(t - 1) + \tilde{\psi}(X(t)).
\]

Fig. 3 shows the histogram of \( \theta(T) \) obtained using the regenerative LSTD, LSTD(0), \( \nabla \cdot \text{LSTD}, \nabla \cdot \text{LSTD}(0) \), and \( \nabla \cdot \text{LSTD}(0.5) \) algorithms, after \( T = 10^5 \) iterations. Observe that, again, the variance of the parameters obtained using the \( \nabla \cdot \text{LSTD} \) algorithms is extremely small compared to the LSTD algorithms.

It is once again noticeable in Fig. 3 that, as before in the results shown in Fig. 2, different values for \( \lambda \) lead to different parameter estimates. To compare performance, the relative Bellman error
was computed

$$\mathcal{E}_B(x) = [P - I]h(x) + c(x) - \eta(T)$$

where $P$ of course depends on the policy $f$, $h = \bar{\theta}^T \psi$, where $\bar{\theta}$ is the mean of the $10^3$ parameter estimates obtained for each of the different algorithms, and $\eta(T)$ denotes the estimate of the average cost $\eta$ using $T = 10^5$ samples. Fig. 4 shows plots of $\mathcal{E}_B(x)$ for each of the five algorithms, for typical values of $\theta(0)$, with $T = 10^3$, $10^4$ and $10^5$. Once again, the feedback policy (51) was used, with $\epsilon = 0.5$.

The Bellman error of the $\nabla$-LSTD algorithms appears to have converged after $T = 10^5$ iterations, and the limit is nearly zero for the range of $x$ where the stationary distribution has nonnegligible mass. Achieving similar performance using the LSTD algorithms requires more than $T = 10^5$ iterations.

VI. CONCLUSION

The new gradient-based TD learning algorithms introduced in this article prove to be excellent alternatives to the more classical TD- and LSTD-learning methods for value function approximation. In the examples considered, the algorithms show remarkable capability to reduce the variance. There are two known explanations for this.

1) The magnitude of the functions that are used as inputs to the $\nabla$-LSTD algorithms is smaller compared to those in the case of LSTD algorithms; for example, if the basis functions for LSTD are polynomials of order $n$, then the basis functions for $\nabla$-LSTD will be of the order $n - 1$.

2) There is an additional “discounting” factor that is inherent in the $\nabla$-LSTD algorithms, due to the derivative sequence $\{A(t)\}$. For example, in the simple linear model experiment (cf. Section V-A), we had $A(t) \equiv a$, for some $a < 0$, and when this term multiplies the original discount factor $\beta$, it can cause a significant reduction in the growth rate of the eligibility trace.

The introduction of $\lambda$ is to further reduce variance. However, the optimal parameter vector obtained using the $\nabla$-LSTD($\lambda$) algorithm will not in general solve the minimum-norm problem (30) if $\lambda < 1$. As in the case of classical LSTD($\lambda$) algorithms, one can expect a bias versus variance tradeoff in choosing $\lambda$ for the $\nabla$-LSTD($\lambda$) algorithms. It is conjectured that as $\lambda \to 0$, the bias becomes larger, and perhaps the variance reduces. A bound similar to (13) is a topic of future research.

Though we only consider problems that involve ultimately estimating the value function for a fixed policy, estimating the gradient of the value function has its own applications.

1) State estimation: In [45], the authors are interested in estimating the gradient of the relative value function, which is useful in obtaining the innovation gain for a nonlinear filter.

2) Control: When one is interested in optimizing the policy using policy iteration in a continuous state space setting, the gradient of the value function could be more useful than the value function, in the policy update step.

3) Mean-field games: As was recently emphasized in [46], “...it is not the Bellman equation that is important, but the gradient of its solution.” That is, it is the gradient of the value functions that is the critical quantity of interest in computation of solutions to mean field games. This appears to indicate that the techniques in this article might offer computational tools for approximating solutions in this class of optimal control problems, and in particular in applications to power systems.

Perhaps the biggest drawback of the $\nabla$-LSTD algorithms is the requirement of the knowledge of $A(t)$ defined in (23). In certain problems (such as in the queuing example discussed in Section V-B) this information is directly available. Other applications may require a combination of system identification with the $\nabla$-LSTD algorithm.

There are many other directions in which this article can be extended. Perhaps the most interesting open question is why the algorithm is so effective even in a discrete state space setting in which there is no theory to justify its application. It will also be worth exploring algorithms analogous to $\nabla$-LSTD, which use finite-differences instead of gradients in a discrete state space setting [50], [51].

We are currently considering the extension of the $\nabla$-LSTD algorithms to a continuous time setting. Though the algorithms are straightforward to obtain, the convergence theory will require extensions of the theory of [28].

Finally, it will be interesting to see how the techniques developed here could be used to estimate the gradient of the state-action value function (either the Q-function of Watkins [47] or SARSA [48]). This will greatly simplify application to control.

APPENDIX

A. Proof of Proposition II.2

Here, we state and prove two simple technical lemmas that are needed for the proof of Proposition II.2. Let $v$ denote the Lyapunov function in Assumption A1.
Lemma A.1: Let $R$ denote a transition kernel that has the Feller property and satisfies, for some $B_0 < \infty$

$$Rv(x) := \int R(x, dy)v(y) \leq B_0v(x), \quad x \in \mathbb{R}^\ell.$$ 

Let $Z$ be a kernel that is absolutely continuous with respect to $R$, with density $\xi : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}$ such that

$$Zg(x) = \int R(x, dy)\xi(x, y)g(y), \quad x \in \mathbb{R}^\ell$$

for any bounded measurable function $g : \mathbb{R}^\ell \to \mathbb{R}$. If the density is continuous and for some $\delta \in (0, 1)$

$$B_\delta := \sup_{x,y} |\xi(x, y)| < \infty$$

then $Z$ has the Feller property: $Zg$ is continuous whenever $g$ is bounded and continuous.

Proof: The proof is based on a truncation argument: Consider the sequence of closed sets

$$S_n = \{x \in \mathbb{R}^\ell : v(x) \leq n\}, \quad n \geq 1.$$ 

Take any sequence of continuous functions $\{\chi_n : n \geq 1\}$ satisfying $0 \leq \chi_n(x) \leq 1$ for all $x$, $\chi_n(x) = 1$ on $S_n$, and $\chi_n(x) = 0$ on $S^c_n$. Hence, $\chi_n$ is a continuous approximation to the indicator on $S_n$.

Denote $g_n = g\chi_n$ for a given bounded and continuous function $g$. The function $Zg_n$ is continuous because $\xi(x, y)g_n(x, y)$ is bounded and continuous. It remains to show that $Zg = \lim_{n\to\infty} Zg_n$, and that the convergence is uniform on compact sets.

Under the assumptions of the lemma, for each $x$

$$|Zg(x) - Zg_n(x)| \leq \|g\|_\infty \int R(x, dy)[1 - \chi_n(y)]|\xi(x, y)|$$

$$\leq B_\delta \|g\|_\infty \int_{S_n} R(x, dy)v_\delta(y).$$

Since $v(y) > n$ on $S^c_n$, this gives, for all $x$

$$|Zg(x) - Zg_n(x)| \leq \frac{1}{n^1-n} B_0 B_\delta \|g\|_\infty v(x).$$

It follows that $Zg_n \to Zg$ uniformly on compact sets, since $v$ is assumed to be continuous.

Lemma A.2: Subject to Assumptions A1 and A2

i) $P^t f$ is continuous if $\|f\|^2 \in L^0_\infty$ and $f$ is continuous.

ii) The vector-valued function $Q^t \nabla f$ is continuous, provided $\nabla f$ is continuous, $\|f\|^2 \in L^0_\infty$, and $\|\nabla f\|^2 \in L^0_\infty$.

Proof: Both parts follow from Lemma A.1, with $R = P^t$. The bound $P_t v \leq B_0 v$ holds under A1, and in fact the constant $B_0$ can be chosen independent of $t$.

For part (i), choose $\xi(x, y) = f(y)$. The Feller property for the kernel $Z$ defined in Lemma A.1 implies in particular that $Zg$ is continuous when $g \equiv 1$. $Zg = P^t f$ in this special case. For part (ii), observe that each $Q^t_{i,j} \leq i, j \leq t$, admits a continuous and bounded density by its definition, cf. (22), (26)

$$Q^t_{i,j}(x, dy) = P^t(x, dy)q^t_{i,j}(x, y).$$

So, we have for each $i$ and $x$

$$[Q^t \nabla f(x)]_i = \int P^t(x, dy)q^t_{i,j}(x, y)[\nabla f(y)]_j.$$ 

Fix $i, j$, and let $\xi(x, y) := q^t_{i,j}(x, y)[\nabla f(y)]_j$. Then Lemma A.1 implies that the $(i, j)$-term in the last sum, $\int P^t(x, dy)q^t_{i,j}(x, y)[\nabla f(y)]_j$, is continuous in $x$.

B. Proof of Lemma III.6

The following shift-operator on sample space is defined for a stationary version of $X$: for a random variable of the form:

$$Z = F(X(r), N(r), \ldots, X(s), N(s)), \quad r \leq s$$

we denote, for any integer $k$

$$\Theta^k Z = F(X(r + k), N(r + k), \ldots, X(s + k), N(s + k)).$$

Consequently, viewing $S(t)$ as a function of $A(1), \ldots, A(t)$ as in the evolution equation (22), we have

$$\Theta^k S(t) = A(t + k) \cdots A(2 + k) A(1 + k). \quad (57)$$

The representation (20) for $\nabla h_{ij}$ is valid under Assumption A3, by Lemma III.3. Using this and (22) gives the first representation in (42)

$$b^T = \frac{1}{E_x \left[(\Omega \nabla c(x))^T \nabla \psi(x)\right] \pi(dx)} \quad \int \beta^T \frac{E_x \left[(\nabla c(x))^T \nabla \psi(x)\right] \pi(dx)}{

\Xi^{\infty} \int \frac{\beta^T E[(\nabla c(X(t))^T \nabla \psi(X(0)))].} \pi(dx)}.$$ 

Stationarity implies that for any $t, k \in Z$

$$E \left[(\Theta^k S(t) \nabla c(X(t)))^T \nabla \psi(X(0))\right]$$

$$= E \left[(\nabla \psi(X(0)))^T (\Theta^t S(t)) \nabla \psi(X(-t))\right].$$ 

Setting $k = -t$, the first representation in (42) becomes

$$b^T = \frac{\Xi^{\infty} \int \frac{\beta^T E[(\nabla c(X(0)))^T (\Theta^{-t} S(t)) \nabla \psi(X(-t))]} \pi(dx)}{

\Xi^{\infty} \int \frac{\beta^T E[(\nabla c(X(0)))^T (\Theta^t S(t)) \nabla \psi(X(-t))]} \pi(dx)}.$$ 

where last equality is obtained under Assumption A3 by applying Fubini’s theorem. This combined with (41) completes the proof.

C. Variance Analysis of $\nabla$-LSTD($\lambda$) Algorithms

As in the case of the classical LSTD($\lambda$) algorithms, the $\nabla$-LSTD($\lambda$) algorithms also belong to a more general class of root-finding algorithms known as stochastic approximation (SA). Following the theory for variance analysis of linear SA recursions, under slightly stronger conditions than Proposition
III.5 and Proposition IV.3, the asymptotic variance of the $\nabla$-LSTD($\lambda$) algorithm is given by the following expression [22], [23], [42]:

$$
\Sigma_\theta := \lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[ \left( \frac{\nabla X(t)}{2} \right)^2 \right] = M^{-1} \Sigma_\Delta \left( M^{-1} \right)^T
$$

where the matrix $M$ is defined in (45), and the “noise covariance matrix” is defined as follows:

$$
\Delta(t) := \tilde{M}(t) \theta^* + \tilde{b}(t)
$$

where (with the quantities defined in Algorithm 2)

$$
\tilde{M}(t) := M - [\nabla \psi(X(t))] - \beta \lambda^T(t + 1) \nabla \psi(X(t + 1)) \zeta(t)
$$

$$
\tilde{b}(t) := b - \zeta^T(t) \psi(X(t))
$$

with $b$ is defined in (46), $\theta^* = M^{-1} b$, and all stochastic processes are assumed stationary. The noise covariance matrix is then expressed by the two equivalent formula

$$
\Sigma_\Delta = \lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[ S(T) S(T)^T \right] = \sum_{t=-\infty}^{\infty} R(t)
$$

where

$$
R(t) = \mathbb{E} \left[ \left( \tilde{M}(t) \theta^* - \tilde{b}(t) \right) \left( \tilde{M}(0) \theta^* - \tilde{b}(0) \right)^T \right], \quad t \geq 0
$$

with $R(-t) = R(t)^T$, and $S(T) = \sum_{t=0}^{T-1} \Delta(t)$. An asymptotic variance formula for LSTD($\lambda$) algorithms can be obtained in a straightforward manner, using the definitions of $\tilde{M}, M(t), b, \text{and } b(t)$ as in Algorithm 1 and Proposition III.1.

In the following, we show how these expressions can be used to calculate the asymptotic variance of the two algorithms when applied to the simple linear model described in Section V-A, and how the algorithms compare with each other with respect to this quantity.

D. Asymptotic Variance for the Linear Model

Consider the application of $\nabla$-LSTD and LSTD algorithms to estimate the value function for the linear model that is analyzed in Section V-A

$$
X(t + 1) = aX(t) + N(t + 1), \quad t \geq 1
$$

with the cost function defined to be a quadratic: $c(x) = x^2$.

The 1-D basis function for $\nabla$-LSTD is given by $\nabla \psi(x) = 2x$, so that the estimate of the value function is $\nabla h_\theta^\psi(x) = 2 \theta x$. In this case, $M$ and $b$ are scalar quantities, and $\varphi \equiv \zeta$ is a scalar sequence

$$
M = 4 \mathbb{E} [X^2(t)] \quad b = 2 \mathbb{E} [X(t) \varphi(t)]
$$

with $\varphi(t)$ defined in (49), and expectations in steady state.

Using the fact that $M \theta = b$, the auto-correlation function $R(t)$ defined in (63) is given by

$$
R(t) = \mathbb{E} \left[ \left( 4 \theta X^2(t) - 2 X(t) \varphi(t) \right) \left( 4 \theta X^2(0) - 2 X(0) \varphi(0) \right) \right].
$$

In the case $\{ N(t) \, : \, t \geq 1 \}$ is i.i.d Gaussian with mean 0 and variance $\sigma_N^2$, using (64), it can be shown that the above expression simplifies to

$$
R(t) = \left( (\theta)^2 + 4 - 4 \theta^* \right) \mathbb{E} \left[ X^2(t) X^2(0) \right] - 2 \theta^* \alpha \beta \sum_{s=0}^{t-1} (a \beta)^s a^{t-s} \mathbb{E} \left[ X^2(s) X^2(s) \right]
$$

where for each $t \geq 1$

$$
\mathbb{E} [X^2(t)] = a^2 \mathbb{E} [X^4(t)] + \left( \sum_{s=0}^{t-1} a^{2s} \right) \sigma_N^2 \mathbb{E} [X^2(t)]
$$

and the steady-state expectations are given by

$$
\mathbb{E} [X^2(t)] := \frac{\sigma_N^2}{1 - \alpha^2}, \quad \mathbb{E} [X^4(t)] := \frac{\sigma_N^2}{1 - \alpha^4} \left( \frac{6}{1 - \alpha^2} + 3 \right)
$$

The noise covariance $\Sigma_\Delta$ is then obtained using the second definition in (62). The plots for theoretical values of the variance $\Sigma_\theta$ in Fig. 1 is obtained using this expression.

Similarly, for the LSTD($\lambda$) algorithm, since $\psi(x) = (1, x^2)^T$, we have

$$
M = \mathbb{E} \left[ \begin{array}{c} 1 \\ X^2(t) \\ X^4(t) \end{array} \right] \quad b = \mathbb{E} \left[ X^2(t) \varphi(t) \right]
$$

with $\varphi(t)$ recursively defined in (48). The auto-correlation can then be obtained using (63), where

$$
\tilde{M}(t) := M - \psi(X(t)) \psi^T(X(t))
$$

$$
\tilde{b}(t) := b - \varphi(t) c(X(t))
$$

and $\theta^* = M^{-1} b$. However, the precise expression for $R(t)$ becomes much more complicated than the one obtained in (65), since it involves calculation of moments up to eighth order.

An alternative way to obtain a theoretical formula for $\Sigma_\Delta$ is using the first definition in (62). For fixed $T$ large enough, one can approximately obtain $\Sigma_\Delta$ by estimating the expectation in (62) using Monte Carlo simulations

$$
\mathbb{E} [S(T) S(T)^T] \approx \frac{1}{N} \sum_{i=0}^{N} S^{(i)}(T) \left( S^{(i)}(T) \right)^T.
$$

The plots for theoretical values of the variance $\Sigma_\theta$ in Fig. 1 were obtained using this approximation.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the Associate Editor for valuable feedback.
REFERENCES

[1] A. Devraj and S. Meyn, “Differential TD learning for value function approximation,” in Proc. IEEE 55th Conf. Decis. Control, 2016, pp. 6347–6354.
[2] D. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming. Cambridge, MA, USA: Athena Scientific, 1996.
[3] S. P. Meyn, Control Techniques for Complex Networks. Cambridge, U.K.: Cambridge Univ. Press, 2007.
[4] S. Chen, A. M. Devraj, A. Bušić, and S. Meyn, “Explicit mean-square error bounds for Monte-Carlo and linear stochastic approximation,” Proc. 23rd Int. Conf. Art. Intell. Statistics, vol. 108, pp. 4173–4183, 2020.
[5] A. M. Devraj, A. Bušić, and S. Meyn, Fundamental design principles for reinforcement learning algorithms, in Handbook on Reinforcement Learning and Control. New York, NY, USA: Springer, 2020.
[6] J. N. Tsitsiklis and B. V. Roy, “Average cost temporal-difference learning,” Automatica, vol. 35, no. 11, pp. 1799–1808, 1999.
[7] D. Bertsekas and S. Shreve, Stochastic Optimal Control: The Discrete-Time Case. Nashua, NH, USA: Athena Scientific, 1996.
[8] V. R. Konda and J. N. Tsitsiklis, “On actor-critic algorithms,” SIAM J. Control Optim., vol. 42, no. 4, pp. 1143–1166, 2003.
[9] V. V. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in Proc. Adv. Neural Inf. Process. Syst., pp. 703–710, 1994.
[10] F. Pineda, “Mean-field theory for batched TD(λ),” Neural Comput., vol. 9, no. 7, pp. 1403–1419, 1997.
[11] H. Yu and D. Bertsekas, “Error bounds for approximations from projected linear equations,” Math. Oper. Res., vol. 35, no. 2, pp. 306–329, 2010.
[12] D. Bertsekas, “Approximate policy iteration: A survey and some new methods,” J. Control Theory Appl., vol. 9, no. 3, pp. 310–335, 2011.
[13] C. Szepesvári, “Least squares temporal difference learning and Galerkin’s method,” Aug. 21–27th 2011, presentation at the Mini-Workshop: Mathematics of Machine Learning, Mathematisches Forschungsinstitut Oberwolfach.
[14] C. Szepesvári, Algorithms for Reinforcement Learning, (Synthesis Lectures on Artificial Intelligence and Machine Learning). San Rafael, CA, USA: Morgan & Claypool, 2010.
[15] J. Boyan, “Least-squares temporal difference learning,” in Proc. Int. Conf. Mach. Learn., 1999, pp. 49–56.
[16] H. J. Kushner and G. G. Yin, Stochastic Approximation Algorithms and Applications, (Applications of Mathematics, vol. 35). New York, NY, USA: Springer-Verlag, 1997.
[17] V. Konda and J. Tsitsiklis, “Actor–critic algorithms,” in Proc. Adv. Neural Inf. Process. Syst., 2000, pp. 1008–1014.
[18] W. Chen et al., “Approximate dynamic programming using fluid and diffusion approximations with applications to power management,” in Proc. 48th IEEE Conf. Decis. Control; 8th Chin. Conf. Decis. Control, 2009, pp. 3575–3580.
[19] T. Yang, P. G. Mehta, and S. P. Meyn, “A mean-field control-oriented approach to particle filtering,” in Proc. Amer. Control Conf., Jul. 2011, pp. 2037–2043.
[20] A. Bensoussan, “Machine learning and control theory (plenary lecture),” in Proc. Adv. Model. Control Power Syst. Future, Palaiseau, France, Sep. 2018.
[21] C. J. C. H. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, Dept. Psychol., King’s College, Cambridge, U.K., 1989.
[22] G. A. Rummery and M. Niranjan, “On-line Q-learning using connectionist systems,” Dept. Eng., Cambridge Univ., Cambridge, U.K. CUED/F-INFENG, Tech. Rep. 166, 1994.
[23] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in Proc. Int. Conf. Mach. Learn., 2014, pp. 387–395.
[24] J. Kiefer and J. Wolfowitz, “Stochastic estimation of the maximum of a regression function,” Ann. Math. Statist., vol. 23, no. 3, pp. 462–466, 1952.
[25] A. Bernstein, Y. Chen, M. Colombino, E. Dall’Anese, P. Mehta, and S. Meyn, “Quasi-stochastic approximation and off-policy reinforcement learning,” in Proc. 55th IEEE Conf. Decis. Control, 2019, pp. 5244–5251.
Adithya M. Devraj received the B.E. degree in telecommunication engineering from the PES Institute of Technology, Bangalore, India, in 2013, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2019.

He is a Postdoctoral Fellow with Stanford University, Stanford, CA, USA, currently conducting research on reinforcement learning with Prof. Benjamin Van Roy. He held a graduate fellowship from the University of Florida Information Institute, Gainesville, FL, USA. He has also held visiting positions with Inria, Paris, France, and the Simon’s Institute for theory of computing, UC Berkeley, Berkeley, CA, USA. He has worked with the Signal Processing for Communication Lab, Indian Institute of Science, Bangalore, India, between 2013 and 2014.

Ioannis Kontoyiannis (Fellow, IEEE) was born in Athens, Greece, in 1972. He received the B.Sc. degree in mathematics from Imperial College (University of London), London, U.K. in 1992, and Part III of the Cambridge University Pure Mathematics Tripos, in 1993, the M.S. degree in statistics and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1997 and 1998, respectively.

Between 1998 and 2001, he was an Assistant Professor with the Department of Statistics, Purdue University, West Lafayette, IN, USA (and also, by courtesy, with the Department of Mathematics, and the School of Electrical and Computer Engineering). From 2000 to 2005, he was an Assistant, then Associate Professor, with the Division of Applied Mathematics and with the Department of Computer Science, Brown University, Providence, RI, USA. Since 2005, he has been a Professor with the Department of Informatics, Athens University of Economics and Business, Athens, Greece. Since 2018, he has been a Professor of Information and Communications with the Department of Engineering, University of Cambridge, Cambridge, U.K., where he is also the Head of the Signal Processing and Communications Laboratory. His research interests include data compression, applied probability, information theory, statistics, simulation, and mathematical biology.

Dr. Kontoyiannis was the recipient of the Manning endowed Assistant Professorship in 2002, and was awarded an honorary Master of Arts degree Ad Eundem, in 2005, both from Brown University. He was the recipient of the Sloan Foundation Research Fellowship in 2004. He has served three terms as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY.

Sean P. Meyn (Fellow, IEEE) received the Ph.D. degree in electrical engineering from McGill University, Montreal, QC, Canada, in 1987.

He held a two year Postdoctoral Fellowship with Australian National University, Canberra, Act, Australia, and was a Professor with the University of Illinois, Champaign, IL, USA, from 1989 to 2011. Since January 2012, he has been a Professor and has held the Robert C. Pittman Eminent Scholar Chair in electrical and computer engineering with the University of Florida, Gainesville, FL, USA. He is the co-author with Richard Tweedie of the monograph Markov Chains and Stochastic Stability, (New York, NY, USA: Springer-Verlag, 1993).

Dr. Meyn also holds an Inria International Chair, Paris, France, from 2019 to 2024. He jointly received the 1994 ORSA/TIMS Best Publication in Applied Probability Award with Tweedie. He and Prof. Bušić share a 2015 Google Research Award: to foster collaboration on demand dispatch for renewable energy integration.