Q-learning and Pontryagin's Minimum Principle

Prashant Mehta
Dept. of Mechanical Science and Engg.
and the Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

Sean Meyn
Dept. of Electrical and Computer Engg.
and the Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

***Abstract*— Q-learning is a technique used to compute an optimal policy for a controlled Markov chain based on observations of the system controlled using a non-optimal policy. It has proven to be effective for models with finite state and action space. This paper establishes connections between Q-learning and nonlinear control of continuous-time models with general state space and general action space. The main contributions are summarized as follows.**

(i) The starting point is the observation that the "Q-function" appearing in Q-learning algorithms is an extension of the Hamiltonian that appears in the Minimum Principle. Based on this observation we introduce the *steepest descent Q-learning* (SDQ-learning) algorithm to obtain the optimal approximation of the Hamiltonian within a prescribed finite-dimensional function class.

(ii) A transformation of the optimality equations is performed based on the adjoint of a resolvent operator. This is used to construct a consistent algorithm based on stochastic approximation that requires only causal filtering of the time-series data.

(iii) Several examples are presented to illustrate the application of these techniques, including application to distributed control of multi-agent systems.

I. Introduction

A. Background

What does Q-learning have to do with the minimum principle? Q-learning is a technique to compute an optimal policy, along with the associated value function, based on observations of the state and input, and without knowledge of the system model. Pontryagin's Minimum Principle is a refinement of the Hamilton-Jacobi-Bellman (HJB) equations that characterize the optimal value function. In this paper we argue that the Hamiltonian appearing in nonlinear control theory is essentially the same as the 'Q-function' that is the object of interest in Q-learning. We find that there is also a close connection between Q-learning and differential dynamic programming [8]. In this way we create a bridge between the reinforcement learning and nonlinear control research communities. In the process we introduce new algorithms for both deterministic and stochastic models.

Several barriers have slowed the creation of a comprehensive theory of reinforcement learning (RL) techniques for models in continuous time, and general state space. In the case of temporal-difference (TD) methods, generalizations to continuous time can be found in the recent works [15] and [13]. The latter is closely related to the linear programming formulation of approximate dynamic programming, which is treated for diffusion models in the recent paper [6]. TD-learning is known to be convergent because it can be interpreted as a stochastic approximation implementation of a steepest descent algorithm applied to a convex optimization problem over a finite dimensional parameter space [14], [3], [1], [10].

In the case of Q-learning the barriers are more fundamental. Watkins introduced the technique in his thesis, and a complete convergence proof appeared later in [17]. An elementary proof based on an associated 'fluid limit model' is contained in [2]. Unfortunately, these results are fragile, depending critically on a finite state space and finite action space. More importantly, these convergence proofs require a complete parameterization that includes all possible Markov models whose state space has a given cardinality. This limits the applicability of these methods because complexity grows with the size of the state space. An extension to general state spaces with finite dimensional parameterization appears in [9], but the convergence result is essentially a local one.

Progress has been more positive for special classes of models. For deterministic linear systems with quadratic cost (the LQR problem), a variant of Q-learning combined with an adaptive version of policy iteration is known to be convergent — see [4] for an analysis in discrete time, and [16] for a similar approach in continuous time. The recent work [12] contains a variant of parameterized Q-learning designed specifically for a class of queueing models.

In this paper we take a fresh look at the algorithm. We formulate a convex optimization problem that characterizes the best approximation of the Q-function within a given class, and from this obtain effective algorithms for on-line computation.

B. Q-learning and the Minimum Principle

To simplify the discussion in this paper we focus attention on a deterministic model in continuous time. Contained in Sec. III is discussion on how the concepts and algorithms introduced in this paper can be extended to more general controlled Markov processes.

The deterministic model is a nonlinear state space model with state process $\boldsymbol{x}$ evolving on X, and input $\boldsymbol{u}$ evolving on U. The state and action space are assumed to be closed subsets of Euclidean space, of dimension ℓ and ℓ_u, respectively. The input-state process is described by the state equation,

$$\tfrac{d}{dt}x(t) = f(x(t), u(t)), \qquad t \geq 0, \tag{1}$$

where $f\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}^\ell$ is assumed to be continuous, and $x(0) \in \mathsf{X}$ is given as an initial condition. Our only assumptions on f and $\boldsymbol{u}$ is that a unique solution exists for each initial condition.

It is assumed that a cost function $c\colon \mathsf{X}\times\mathsf{U} \to \mathbb{R}_+$ is given, and we let J^* denote an associated value function. In this section and throughout most of the paper we restrict to the discounted cost optimality criterion. For a given discount rate $\gamma > 0$, the value function is defined by

$$J^*(x) = \inf \int_0^\infty e^{-\gamma s} c(x(s), u(s))\, ds, \qquad x(0) = x, \quad (2)$$

where the infimum is over all $\boldsymbol{u}$.

We will borrow concepts from Markov decision theory and the theory of general state space Markov processes. In particular, the *generator* that appears in analysis of Markov processes is a first order differential operator in this special case. For any C^1 function $h\colon \mathsf{X} \to \mathbb{R}$ we define the function of two variables,

$$\mathcal{D}_u h\,(x) := (\nabla h\,(x))^{\intercal} f(x,u), \qquad (x,u) \in \mathsf{X} \times \mathsf{U}. \quad (3)$$

The generator is defined so that the function $f_h(x,u) := \mathcal{D}_u h\,(x)$ satisfies $\frac{d}{dt} h(x(t)) = f_h(x(t), u(t))$ for any state-input pair $(\boldsymbol{x}, \boldsymbol{u})$. In the remainder of this paper, ∇ is reserved to denote the gradient operator with respect to x.

If the value function (2) is finite valued, then the discounted-cost optimality equation holds,

$$\min_u \big(c(x,u) + \mathcal{D}_u J^*\,(x)\big) = \gamma J^*(x), \qquad x \in \mathsf{X}. \quad (4)$$

The most natural analog of the "Q-function" that appears in Q-learning is the function of two variables within the minimum on the left hand side of (4). We denote this by,

$$H^*(x,u) = c(x,u) + \mathcal{D}_u J^*\,(x), \qquad (x,u) \in \mathsf{X} \times \mathsf{U}. \quad (5)$$

The function H^* will be the object of study throughout the paper. This function appears in the method of *differential dynamic programming* [8]. By consideration of $H^*(x,u)$ in place of $J^*(x)$, it is shown in [8] that algorithms can be constructed to obtain an optimal policy *locally*, in a neighborhood of some nominal trajectory. The goal here is to find an *approximation* to the optimal policy that is defined everywhere.

The Q-function (5) is closely related to the Hamiltonian. Recall that the Hamiltonian is expressed,

$$H^*(x,u;p) = c(x,u) + p^{\intercal} f(x,u)$$

where $p \in \mathbb{R}^\ell$ is the costate variable [8]. Under appropriate assumptions on c and f we have the Minimum Principle: For each initial condition there is an optimal state trajectory $\boldsymbol{x}^*$ together with an optimal *costate* trajectory $\boldsymbol{p}^*$ such that the Hamiltonian $H^*(x^*(t), u; p^*(t))$ is minimized over $u \in \mathsf{U}$ when $u = u^*(t)$ is optimal.

It is known that, under appropriate conditions, the costate trajectory can be expressed $p^*(t) = \nabla J\,(x^*(t))$. On substitution we conclude that $H^*(x,u;p^*) = H^*(x,u)$ when $x = x^*(t)$ and $p^* = p^*(t)$.

For the discounted-cost optimality criterion, the minimum of the Q-function is given by the scaled value function,

$$\underline{H}^*(x) := \min_{u\in\mathsf{U}} H^*(x,u) = \gamma J^*(x). \quad (6)$$

Substituting into (5) then gives a fixed point equation

$$\mathcal{D}_u \underline{H}^*\,(x) = -\gamma(c(x,u) - H^*(x,u)). \quad (7)$$

This is essentially equivalent to the fixed point equation that appears in Q-learning, and this is the promised bridge between Q-learning and the Minimum Principle. The goal of the paper is to show how methods from Q-learning and TD-learning can be adapted to construct optimal approximations of the Q-function $H^*(x,u)$.

C. Contributions and overview

The contributions of the paper are summarized as follows:

(i) The recognition of the connections between Q-learning, differential dynamic programming, and the Minimum Principle allows us to unify and extend previous results on adaptive control for *deterministic* nonlinear systems.

(ii) The central idea of TD-learning is to construct a causal representation of the value function to facilitate the creation of a causal on-line algorithm. In [10, Ch. 11] it is argued that this step can be regarded as the application of an adjoint operator in a certain Hilbert space. The algorithms introduced here are based on the extension of this idea to Q-learning.

(iii) The application to distributed control of multi-agent systems, described with the aid of an example in Sec. IV-C, is remarkably successful, given the very partial theoretical support for this approach. These numerical results invite many avenues for future research.

The remainder of the paper is organized as follows. The next section develops the theory of Q-learning for deterministic systems. The final refinement is contained in Sec. II-D, where the convex representation of the Q-function is introduced. Extensions to Markov models are discussed in Sec. III, several examples are presented in Sec. IV, and the paper ends with conclusions and suggestions for future research in Sec. V.

II. Q-LEARNING FOR DETERMINISTIC MODELS

In this section we develop methods to construct approximate solutions to the fixed point equation (7). Extensions to the total cost optimization problem are described in Sec. II-E.

It is assumed that we have a parameterized family of real-valued functions on $\mathsf{X}\times\mathsf{U}$, denoted $\{H^\theta(x,u) : \theta \in \mathbb{R}^d,\ x \in \mathsf{X},\ u \in \mathsf{U}\}$. Our goal is to choose the parameter θ so that $H^\theta \approx H^*$, where the approximation is with respect to a specific norm chosen for ease of computation *and* to weight state-input pairs according to their relative importance.

We begin with the specification of an error criterion.

A. Bellman error

Throughout the paper we adopt a Hilbert space setting for approximation. It is assumed that a probability distribution ϖ on $\mathcal{B}(\mathsf{X} \times \mathsf{U})$ is given, and for any square-integrable measurable functions $F, G\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$ we denote an inner product and norm through the definitions,

$$\begin{aligned} \langle F, G \rangle_\varpi &:= \int F(x,u) G(x,u)\, \varpi(dx, du), \\ \|F\|^2_\varpi &:= \int F^2(x,u)\, \varpi(dx, du). \end{aligned} \tag{8}$$

We let $L_2(\varpi)$ denote those functions with finite norm. This function space is a Hilbert space with inner-product $\langle \cdot\,, \cdot \rangle_\varpi$.

The most natural error criterion based on the Hilbert space norm (8) is the direct error $\|H^* - H^\theta\|_\varpi$. This is not easily minimized since H^* is not known. An alternative is the Bellman error, defined as the mean-square error in the fixed point equation. For the discounted-cost setting this is expressed,

$$\mathcal{E}_{\text{Bell}}(\theta) := \tfrac{1}{2} \|\mathcal{L}^\theta\|^2_\varpi, \qquad \theta \in \mathbb{R}^d \tag{9}$$

where the point-wise error is defined by

$$\mathcal{L}^\theta(x,u) := \mathcal{D}_u \underline{H}^\theta\,(x) + \gamma(c - H^\theta) \tag{10}$$

with $\underline{H}^\theta(x) := \inf_u H^\theta(x,u)$. If the infimum is achieved for each x then we denote,

$$\phi^\theta(x) \in \arg\min_u H^\theta(x,u), \qquad x \in \mathsf{X}. \tag{11}$$

Our goal is to minimize the Bellman error over all $\theta \in \mathbb{R}^d$. If the Bellman error is zero for some θ^*, and if the support of ϖ is full, then the policy ϕ^{θ^*} is optimal.

We let ψ^θ denote the gradient of H^θ with respect to θ. Assuming that we can take the derivative under the norm, the optimal parameter solves the nonlinear equation,

$$\langle \mathcal{L}^\theta, \mathcal{D}_u \underline{\psi}^\theta_i - \gamma \psi^\theta_i \rangle_\varpi = 0, \tag{12}$$

with $\underline{\psi}^\theta_i(x) = \psi^\theta_i(x, \phi^*(x))$, $1 \le i \le d$. As with $\underline{H}$ in (6), the underline notation is used to denote the minimum taken with respect to $u \in \mathsf{U}$ and $\phi^*(x)$ is the minimizing policy.

A solution to (12) can be found using steepest descent: Let $\boldsymbol{\theta}$ denote the solution to the differential equation,

$$\tfrac{d}{dt}\theta = -\nabla \mathcal{E}_{\text{Bell}}(\theta) = -\langle \mathcal{L}^\theta, \mathcal{D}_u \underline{\psi}^\theta - \gamma \psi^\theta \rangle_\varpi \tag{13}$$

The right hand side appears to depend upon the unknown dynamics, since $\mathcal{D}_u$ is defined based on the nonlinear system equations (1). This is resolved by recalling the interpretation of the generator as representing the time derivative. For example,

$$\mathcal{D}_u \underline{H}^\theta\,(x) = \tfrac{d}{dt} \underline{H}^\theta(x(t)) \Big|_{\substack{x = x(t) \\ u = u(t)}}$$

Consequently, a version of the ODE (13) can be constructed so that the right hand side is computable without knowledge of system dynamics.

There are however two issues that must be overcome: First is the complexity of integration over the state-input space, which is required in the computation of the inner-product appearing in (13). Second is the differentiation of potentially noisy measurements. The first issue is overcome in Sec. II-B by constructing a stationary solution to the deterministic model. The second issue is addressed in Sec. II-C by integrating the fixed point equation, and performing a transformation to obtain a causal characterization of H^* that does not involve a time-derivative. The steps taken to create this representation are adapted from TD-learning techniques [1], [10].

For the purposes of approximation we choose an input $\boldsymbol{u}$ so that the Law of Large Numbers (LLN) holds with respect to ϖ, in the following form: For each continuous function $F\colon \mathbb{R}^\ell \times \mathbb{R}^{\ell_u} \to \mathbb{R}$ that is integrable with respect to ϖ, and almost every initial condition $(x(0), u(0))$, we have as $T \to \infty$,

$$\frac{1}{T} \int_0^T F(x(t), u(t))\, dt \longrightarrow \int_{\mathsf{X} \times \mathsf{U}} F(x,u)\, \varpi(dx, du). \tag{14}$$

For example, under general conditions on the system (1) with a scalar input, the input $\boldsymbol{u}$ can be chosen as a linear combination of sinusoids.

The creation of an ergodic realization of the nonlinear model is what allows us to construct computationally efficient on-line algorithms to compute an optimal approximation.

B. A stationary environment for learning

The usual Q-learning algorithm is based on the assumption that a randomized stationary policy is applied [1]. Randomization is imposed to allow sufficient sampling of the state-input space, very much like persistence of excitation is required in adaptive control algorithms. We impose a similar set of assumptions here.

In the deterministic model the "excitation signal" is assumed to be deterministic: Suppose that $\boldsymbol{\xi}$ is an exogenous process evolving on a subset of ℓ_w-dimensional Euclidean space, denoted W. The control at time t is a function of the state and this process,

$$u(t) = \phi(x(t), \xi(t)), \qquad t \ge 0, \tag{15}$$

for some function $\phi\colon \mathsf{X} \times \mathsf{W} \to \mathsf{U}$.

The following assumptions are imposed in our treatment of the deterministic model:

(A1) The excitation process is the solution to an autonomous ODE: For a Lipschitz-continuous function $g\colon \mathsf{W} \to \mathbb{R}^{\ell_w}$,

$$\tfrac{d}{dt}\xi(t) = g(\xi(t)). \tag{16}$$

(A2) There exists a probability measure Γ on $\mathcal{B}(\mathsf{X} \times \mathsf{W})$ such that the joint process is stationary on the non-negative time-axis when $(x(0), \xi(0)) \sim \Gamma$. It is ergodic in the sense that the LLN holds for every continuous function $F\colon \mathbb{R}^\ell \times \mathbb{R}^{\ell_w} \to \mathbb{R}$ and almost every initial condition $(x(0), \xi(0)) \in \mathsf{X} \times \mathsf{W}$: As $T \to \infty$,

$$\frac{1}{T} \int_0^T F(x(t), \xi(t))\, dt \longrightarrow \int F(x,w)\, \Gamma(dx, dw). \tag{17}$$

The support of Γ is a compact subset of $\mathbb{R}^{\ell} \times \mathbb{R}^{\ell_w}$.

Under (A1) and (A2) we let $(\boldsymbol{x}^\circ, \boldsymbol{\xi}^\circ)$ denote the stationary version of the process on the two-sided time interval, with marginal Γ, and we denote $u^\circ(t) = \phi(x^\circ(t), \xi^\circ(t))$.

Under these assumptions we *define* the induced marginal ϖ as follows: For any measurable function $g\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$,

$$\int g(x,u)\, \varpi(dx, du) = \int g(x, \phi(x,w))\, \Gamma(dx, dw).$$

Observe that $(\boldsymbol{x}^\circ, \boldsymbol{u}^\circ)$ is stationary with marginal ϖ, and the LLN (14) holds for a.e. initial condition $[\varpi]$.

The joint process $(x(t), \xi(t))$ is the solution to an autonomous differential equation under these assumptions, $\frac{d}{dt}(x(t), \xi(t)) = (f(x, \phi(x(t)), \xi(t))), g(\xi(t)))$. For any continuously differentiable function $h\colon \mathsf{X} \times \mathsf{W} \to \mathbb{R}$ we denote,

$$\mathcal{D}h\,(x,w) = \frac{\partial}{\partial x} h(x,w) \cdot f(x, \phi(x,w)) + \frac{\partial}{\partial w} h(x,w) \cdot g(w)$$

This is the generator for the joint process, defined so that for any time t,

$$\mathcal{D}h\,(x,w) = \tfrac{d}{dt} h(x(t), \xi(t))\Big|_{\substack{x=x(t)\\ w=\xi(t)}}. \tag{18}$$

This definition is consistent with (3): If h is a function of x alone then,

$$\tfrac{d}{dt} h(x(t))\Big|_{\substack{x=x(t)\\ w=\xi(t)}} = \mathcal{D}_u h\,(x), \quad \textit{when } u = \phi(x,w).$$

We can now define the SDQ-learning algorithm as an approximation of the steepest descent algorithm to minimize (9):

$$\tfrac{d}{dt}\theta = -\varepsilon \langle \mathcal{L}^\theta, \mathcal{D}_u \nabla_\theta \underline{H}^\theta - \gamma \nabla_\theta H^\theta \rangle_\varpi \tag{19}$$

where $\varepsilon = \varepsilon(\theta(t), t)$ is a positive gain. In the stochastic approximation approach the inner product (defined as an expectation) is replaced by realized values,

$$\tfrac{d}{dt}\theta = -\varepsilon_t \mathcal{L}_t^\theta \big(\tfrac{d}{dt} \nabla_\theta \underline{H}^\theta\,(x^\circ(t)) - \gamma \nabla_\theta H^\theta(x^\circ(t), u^\circ(t)) \big), \tag{20}$$

where $\mathcal{L}_t^\theta$ is realized as

$$\mathcal{L}_t^\theta := \tfrac{d}{dt} \underline{H}^\theta\,(x^\circ(t)) + \gamma (c(x^\circ(t), u^\circ(t)) - H^\theta(x^\circ(t), u^\circ(t))). \tag{21}$$

To justify this representation we impose further assumptions:

(A3) $H^\theta(x,u)$ and $\underline{H}^\theta(x)$ are C^1 functions of (x, u, θ).

Example. Consider the special case in which the dynamics are linear in u, and the cost quadratic in u:

$$\begin{aligned} \tfrac{d}{dt} x(t) &= f(x(t)) + g(x(t)) u(t) \\ c(x,u) &= c(x) + \tfrac{1}{2} u^{\intercal} R u \end{aligned} \tag{22}$$

with $c\colon \mathsf{X} \to \mathbb{R}_+$ and $R > 0$. Since dynamics are linear in u, we consider Q-function of the form $H^*(x,u) = c(x,u) + E^*(x) + u^{\intercal} F^*(x)$. The minimization over U in (6) gives

$$\phi^*(x) = -R^{-1} F^*\,(x), \tag{23}$$

and we have,

$$\underline{H}^*(x) = H(x, \phi^*(x)) = c(x) + E^*(x) - \tfrac{1}{2} F^*(x)^{\intercal} R^{-1} F^*(x). \tag{24}$$

For Q-learning, a parameterization $H^\theta(x,u) = c(x,u) + E^{\theta^{\mathrm{x}}}(x) + u^{\intercal} F^{\theta^{\mathrm{xu}}}(x)$ is considered in terms of a *given* set of basis functions, scalar valued functions $\{\psi_i^{\mathrm{x}}\}$ and vector-valued functions $\{\psi_i^{\mathrm{xu}}\}$:

$$H^\theta(x,u) = c(x,u) + \tfrac{1}{2} \sum_{i=1}^{d_{\mathrm{x}}} \theta_i^{\mathrm{x}} \psi_i^{\mathrm{x}}(x) + \sum_{j=1}^{d_{\mathrm{xu}}} \theta_j^{\mathrm{xu}} u^{\intercal} \psi_j^{\mathrm{xu}}(x). \tag{25}$$

In this notation the minimum of H^θ over u is given by,

$$\underline{H}^\theta(x) = c(x) + E^{\theta^{\mathrm{x}}}(x) - \tfrac{1}{2} F^{\theta^{\mathrm{xu}}}(x)^{\intercal} R^{-1} F^{\theta^{\mathrm{xu}}}(x). \tag{26}$$

The function $\underline{H}^\theta(x)$ is linear in θ^{x}, and quadratic in θ^{xu}.

A version of the steepest descent algorithm (19) is given by,

$$\begin{aligned} \tfrac{d}{dt}\theta_i^{\mathrm{x}}(t) &= -\varepsilon^{\mathrm{x}} \langle \mathcal{L}^\theta, \mathcal{D}_u \psi_i^{\mathrm{x}} - \gamma \psi_i^{\mathrm{x}} \rangle_\varpi \\ \tfrac{d}{dt}\theta_i^{\mathrm{xu}}(t) &= -\varepsilon^{\mathrm{xu}} \langle \mathcal{L}^\theta, -\mathcal{D}_u [F^{\theta^{\mathrm{xu}\,\intercal}} R^{-1} \psi_i^{\mathrm{xu}}] - \gamma \psi_i^{\mathrm{xu}} \rangle_\varpi \end{aligned}$$

The SDQ-learning algorithm is the stochastic approximation of these equations obtained via (20). □

Our next task is to remove the derivative in (21).

C. Causal smoothing without bias

The value function J^* is defined as an integral of discounted future cost. The main idea of TD learning is to obtain a representation of a value function in terms of integrals with respect to the reversed-time process. This construction can be performed based on a resolvent kernel and its adjoint [10].

The resolvent acts on functions $g\colon \mathsf{X} \times \mathsf{W} \to \mathbb{R}$ via

$$R_\beta g\,(x,w) = \int_0^\infty e^{-\beta t} g(x(t), \xi(t))\, dt, \tag{27}$$

with initialization $x(0) = x$, $\xi(0) = w$. The following resolvent equation is central: We have for any $\beta > 0$ and any function g for which $g^\beta := R_\beta g$ is finite valued,

$$\mathcal{D} g^\beta = \beta g^\beta - g \tag{28}$$

This follows from the interpretation of the generator in (18), but is valid even if g^β is not everywhere differentiable. It is straightforward to show that the integral form always holds,

$$g_T^\beta - g_0^\beta = \int_0^T \big(\beta g_t^\beta - g_t \big)\, dt, \quad T > 0,\ (x,w) \in \mathsf{X} \times \mathsf{W}\,,$$

where $h_t = h(x(t), \xi(t))$ for any $h\colon \mathsf{X} \times \mathsf{W} \to \mathbb{R}$.

The adjoint $R_\beta^\dagger$ can be expressed in terms of the stationary process $(\boldsymbol{x}^\circ, \boldsymbol{\xi}^\circ)$.

Proposition 2.1: The following identities hold for any $\beta > 0$, and any functions $g, h \in L_2(\Gamma)$:

(i) R_β is a bounded linear operator from $L_2(\Gamma)$ to $L_2(\Gamma)$, with induced operator norm $\|R_\beta\|_\Gamma \le \beta^{-1}$.

(ii) Its adjoint coincides with the resolvent for the time-reversed process,

$$R_\beta^\dagger g\,(x,w) = \int_0^\infty e^{-\beta t} \mathsf{E}_{x,\,w}[g(x^\circ(-t), \xi^\circ(-t))]\, dt$$

with expectation conditional on $x^\circ(0) = x$, $\xi^\circ(0) = w$.

(iii) $R_\beta^\dagger R_\beta = (2\beta)^{-1}(R_\beta^\dagger + R_\beta)$. That is,

$$\langle R_\beta g, R_\beta h\rangle = \frac{1}{2\beta}\big(\langle g, R_\beta^\dagger h\rangle + \langle h, R_\beta^\dagger g\rangle\big)$$

Proof: Part (i) follows from Jensen's inequality. See [10, Ch. 11] for a proof of (ii) in the Markovian setting. Part (iii) is established using elementary calculus. Here is an informal interpretation based on the following two facts: The generator for the time-reversed process is $-\mathcal{D}$, and we have $R_\beta = [\beta I - \mathcal{D}]^{-1}$ on a suitable domain. Consequently,

$$R_\beta^\dagger R_\beta = [\beta I + \mathcal{D}]^{-1}[\beta I - \mathcal{D}]^{-1}$$

so that (iii) can be interpreted as a partial fraction expansion. □

To define the SDQ(β)-learning algorithm we first obtain a steepest descent algorithm based on a transformation of the dynamic programming equation. For a given $\beta > 0$ we denote $\mathcal{L}^{\theta,\beta} = R_\beta \mathcal{L}^\theta = [\beta R_\beta - I]\underline{H}^\theta + \gamma R_\beta(c - H^\theta)$, which is a function of the initial condition $(x(0), \xi(0))$. If this function is zero for some θ and all (x, w), it then follows from the resolvent equation that the same is true flor $\mathcal{L}^\theta$.

The smoothed Bellman error is given by $\mathcal{E}_\beta(\theta) := \frac{1}{2}\|\mathcal{L}^{\theta,\beta}\|_\varpi^2$, whose gradient can be expressed using Proposition 2.1 in terms of the adjoint,

$$\begin{aligned}\nabla\mathcal{E}_\beta(\theta) &= \langle \mathcal{L}^{\theta,\beta}, \nabla_\theta \mathcal{L}^{\theta,\beta}\rangle_\varpi \\ &= \langle \underline{H}^\theta, \nabla_\theta \underline{H}^\theta\rangle_\varpi - \Big(\langle R_\beta^\dagger \underline{H}^\theta, \zeta\rangle_\varpi + \langle d, R_\beta^\dagger \nabla_\theta \underline{H}^\theta\rangle_\varpi\Big) \\ &\qquad + \frac{1}{2\beta}\Big(\langle R_\beta^\dagger d, \zeta\rangle_\varpi + \langle d, R_\beta^\dagger \zeta\rangle_\varpi\Big)\end{aligned} \tag{29}$$

with $d(x,u) := \gamma(c(x,u) - H^\theta(x,u)) + \beta \underline{H}^\theta(x)$, and $\zeta(x,u) := [-\gamma\nabla_\theta H^\theta + \beta\nabla_\theta \underline{H}^\theta]$.

Each of these five inner products can be estimated using sample path averages. The only complication is the adjoint $R_\beta^\dagger$ appearing in four of these terms. To treat this we borrow from TD learning.

Consider the final inner product $\langle d, R_\beta^\dagger \zeta\rangle_\varpi$ appearing in (29). Denote by $\boldsymbol{\zeta}^\beta = \{\zeta_t^\beta : t \ge 0\}$ the process obtained by filtering $\boldsymbol{\zeta} := \{\zeta_s = \zeta(x^\circ(s), u^\circ(s))\}$,

$$\zeta_t^\beta = \int_0^t e^{-\beta(t-s)}\zeta_s\, ds.$$

$\boldsymbol{\zeta}^\beta$ is similar to the *eligibility vector* that appears in TD-learning. The LLN gives, for a.e. initial condition,

$$\lim_{T\to\infty} \frac{1}{T}\int_0^T d_t \zeta_t^\beta\, dt = \langle d, R_\beta^\dagger \zeta\rangle_\varpi.$$

Based on these representations we arrive at the SDQ(β)-learning algorithm — a stochastic approximation of steepest descent,

$$\begin{aligned}\tfrac{d}{dt}\theta = -\varepsilon_t\Big[\underline{H}_t^\theta \nabla_\theta \underline{H}_t^\theta - \Big((\underline{H}^\theta)_t^\beta \zeta_t + d_t(\nabla_\theta \underline{H}^\theta)_t^\beta\Big) \\ + \frac{1}{2\beta}\Big(d_t^\beta \zeta_t + d_t \zeta_t^\beta\Big)\Big].\end{aligned} \tag{30}$$

An approximate Newton-Raphson algorithm can be obtained in a manner similar to LSTD learning [10]. However, the motivation is not clear since the Bellman error is not convex in θ. We now introduce a convex setting to obtain an approximate Newton-Raphson algorithm.

D. A convex characterization of the Q-function

Our goal in this section is to find a fixed point equation that characterizes an optimal approximation to the Bellman equation, obtained as a stationary point for a convex program. We seek an approximation among the affine family,

$$H^\theta(x,u) = c(x,u) + \theta^{\mathrm{T}}\psi(x,u), \quad x \in \mathsf{X},\ u \in \mathsf{U}. \tag{31}$$

Even in this special case, the Bellman error $\mathcal{E}_{\text{Bell}}$ is not a convex function of θ, so that the existence of a unique global minimum is unresolved. Specifically, the difficulty is that $\underline{H}^\theta(x)$ is not an affine function of θ. To frame the approximation problem in a convex analytic setting we first take a second look at the standard linear programming approach to dynamic programming.

The discounted cost optimization problem can be formulated as the infinite-dimensional linear program (LP) over the space of C^1 functions $J\colon \mathsf{X} \to \mathbb{R}_+$:

$$\begin{aligned}\max\quad & \langle 1, J\rangle_\varpi \\ \text{s.t.}\quad & c(x,u) + \mathcal{D}_u J\,(x) \ge \gamma J(x) \quad \text{all } x,\ u\end{aligned} \tag{32}$$

where $\langle 1, J\rangle_\varpi = \int J(x)\varpi(dx, du)$ is the steady-state mean of J. If we enlarge the variable space to include functions of two variables (x, u) then we obtain in (33) an LP that characterizes H^*,

$$\begin{aligned}\max\quad & \gamma^{-1}\langle 1, G\rangle_\varpi \\ \text{s.t.}\quad & c(x,u) + \gamma^{-1}\mathcal{D}_u G\,(x) \ \ge\ H(x,u) \\ & H(x,u) \ \ge\ G(x).\end{aligned} \tag{33}$$

The variables in this LP are the pair of functions (G, H), with G in the domain of the generator. The inequality constraints in (33) can be interpreted as a relaxation of the identity (5).

Note that in this non-parameterized setting, the first inequality constraint in (33) can be taken to be an equality constraint without loss of generality in any optimizer. The following result, stated without proof, easily implies that an optimizer (G^*, H^*) for (33) will satisfy $\gamma^{-1}G^*(x) = J^*(x)$ for a.e. x $[\varpi]$.

Lemma 2.2: The following are equivalent for a pair of functions $G\colon \mathsf{X} \to \mathbb{R}$, $H\colon \mathsf{X}\times\mathsf{U} \to \mathbb{R}$, with G in the domain of the generator:

(i) $\mathcal{D}_u G\,(x) = -\gamma(c(x,u) - H(x,u))$ and $G(x) \le H(x,u)$ for each (x,u).

(ii) There exists a function $c^-\colon \mathsf{X}\times\mathsf{U} \to \mathbb{R}$ satisfying $c^- \le c$ everywhere,

$$\begin{aligned}H(x,u) - c(x,u) &= H^-(x,u) - c^-(x,u) \\ G(x) &= \underline{H}^-(x), \qquad x \in \mathsf{X},\ u \in \mathsf{U},\end{aligned}$$

with $\underline{H}^-(x) := \min_{u'} H^-(x,u')$, and H^- the Q-function associated with c^-.

□

These results suggest many approaches to approximating H^* within the affine family (31). Suppose that $\{(G^\theta, H^\theta) :$

$\theta \in \mathbb{R}^d\}$ is a parameterized family of functions. Consider a variant of the smoothed error $\mathcal{L}^{\theta,\beta}$ defined in Sec. II-C,

$$\mathcal{L}^{\theta,\beta} = [\beta R_\beta - I]G^\theta + \gamma R_\beta(c - H^\theta) \tag{34}$$

Our goal is to maximize $\langle 1, G^\theta\rangle_\varpi$, subject to $\mathcal{L}^{\theta,\beta}(x,u) = 0$ and $H(x,u) \geq G(x)$ for all x, u. We can construct a convex loss-function by relaxing these constraints through a penalty function. For fixed $\kappa > 0$ define,

$$\mathcal{E}_{\text{Bell-LP}}(\theta) = -\langle 1, G^\theta\rangle_\varpi + \frac{\kappa}{2}\big(\|(G^\theta - \underline{H}^\theta)_+\|_\varpi^2 + \|\mathcal{L}^{\theta,\beta}\|^2\big) \tag{35}$$

The function of θ given by $(G^\theta(x) - \underline{H}^\theta(x))_+^2$ is convex for each x when the parameterization is affine. The loss-function $\mathcal{E}_{\text{Bell-LP}}(\theta)$ is also convex. We can define a steepest descent or Newton algorithm to compute the value θ^* that achieves its minimum. An on-line implementation is then created by applying the adjoint formulae as in the previous algorithms.

E. Total cost criterion

Extension to the total cost criterion is straightforward. The total cost value function is given by (2) with $\gamma = 0$, and in this case the optimality equation becomes,

$$\min_u \big(c(x,u) + \mathcal{D}_u J^*(x)\big) = 0, \qquad x \in \mathsf{X}.$$

Consequently, with H^* defined in (5) we obtain a version of the fixed point eq. (6), $\underline{H}^*(x) = \min_u H^*(x,u) = 0$. This unfortunately does not provide a useful fixed point equation for application of Q-learning techniques.

Consider the modified definition: For a given $\varrho > 0$ redefine H^* by,

$$H^*(x,u) = \varrho^{-1} J^*(x) + c(x,u) + \mathcal{D}_u J^*(x) \tag{36}$$

The fixed point equation obtained from the dynamic programming equation is then

$$\underline{H}^*(x) = \min_u H^*(x,u) = \varrho^{-1} J^*(x), \qquad x \in \mathsf{X}.$$

Substituting $J^* = \varrho^{-1}\underline{H}^*$ into (36) then gives,

$$\mathcal{D}_u \underline{H}^*(x) = -\varrho[c(x,u) - H^*(x,u) + \underline{H}^*(x)] \tag{37}$$

Any of any of the algorithms introduced in this section can be adapted to approximate H^* by using (37) in place of (7).

III. Extensions to Markov models

We now describe briefly extension to Markov models. It is assumed that $\boldsymbol{\xi}$ is itself Markovian, and that the joint process $(\mathbf{x}, \boldsymbol{\xi})$ is Harris ergodic [11]. The resolvent is then defined as in (27), with the inclusion of a conditional expectation

$$R_\beta g(x,w) = \int_0^\infty e^{-\beta t} \mathsf{E}_{x,w}[g(x(t), \xi(t))]\, dt, \tag{38}$$

conditional on $x^\circ(0) = x$ and $\xi^\circ(0) = w$. The (extended) generator for the joint process is virtually *defined* by the resolvent equation (28). Consequently, we can almost obtain the expression (29) for the gradient of the Bellman error. Unfortunately the development of Sec. II-C fails at one crucial point.

A. Causal smoothing fails for Bellman error

The final term in (29) was obtained from the partial fraction representation $R_\beta^\dagger R_\beta = (2\beta)^{-1}(R_\beta^\dagger + R_\beta)$ stated in Proposition 2.1. In the probabilistic setting the adjoint of the generator is not simply the negative, and this representation fails. To obtain a convex program to define an optimal approximation we return to the original Q-learning formalism.

B. Galerkin relaxation

Our goal is to find H^θ so that the Bellman error (9) is zero, or nearly so. Suppose that instead we insist that its projection on a subspace is zero.

Let φ denote a d-dimensional function on $\mathsf{X} \times \mathsf{W}$, and choose θ so that the projection onto the span of $\{\varphi_i\}$ is zero:

$$0 = \langle \mathcal{L}^{\theta,\beta}, \varphi_i\rangle_\varpi, \qquad 1 \leq i \leq d. \tag{39}$$

The starting point of the Q-learning formalism is to interpret this as a stationary point for the ODE,

$$\tfrac{d}{dt}\theta = -\langle \mathcal{L}^{\theta,\beta}, \varphi\rangle_\varpi$$

Under general conditions, this ODE is locally asymptotically stable [9]. In particular, it is assumed in [9] that the parameterization is linear, and that $\varphi \equiv \psi$. The ODE is known to be *globally* asymptotically stable when these conditions hold, and in addition $\{\psi_i\}$ consists of indicator functions of sets that form a partition of $\mathsf{X} \times \mathsf{U}$ [1].

A consistent algorithm is obtained by applying the convex characterization of Sec. II-D. The optimal discounted cost remains a solution to (33) under general conditions on the Markov model.

Let $\mathcal{L}^{\theta,\beta}$ denote the smoothed error defined in (34). Exactly as in (35) we obtain a convex program through the introduction of penalty functions:

$$\begin{aligned}\mathcal{E}_{\text{Bell-G}}(\theta) = -\langle 1, G^\theta\rangle_\varpi + \frac{\kappa}{2}\|(G^\theta - \underline{H}^\theta)_+\|_\varpi^2 \\ + \frac{\kappa}{2}\sum_{i=1}^d \big(\langle \mathcal{L}^{\theta,\beta}, \varphi_i\rangle_\varpi\big)^2\end{aligned} \tag{40}$$

where $\kappa > 0$ is fixed. We can define a steepest descent or Newton algorithm to compute the value θ^* that achieves its minimum. This can be translated to form a stochastic approximation algorithm using standard techniques, along with the introduction of the adjoint to create a causal algorithm.

IV. Examples

The examples considered here are all taken to be the special case of (22) with a cost quadratic in u.

A. Local approximation for a nonlinear system

The following simple example is borrowed from [8]:

$$\tfrac{d}{dt}x = -x^3 + u, \qquad c(x,u) = \tfrac{1}{2}x^2 + \tfrac{1}{2}u^2 \tag{41}$$

The HJB equation (22) and the formula (23) give $\phi^*(x) = -\nabla J^*(x)$ and

$$\min_u \{\tfrac{1}{2}x^2 + \tfrac{1}{2}u^2 + (-x^3 + u)\nabla J^*(x)\} = \gamma J^*(x).$$

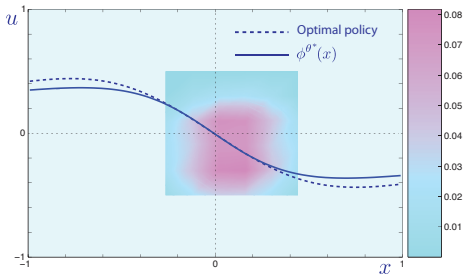

Fig. 1: Comparison of the optimal policy and the policy obtained from H^{θ^*} based on SDQ(γ) for the scalar example (41). Also shown is a density plot of the stationary distribution ϖ. The approximation is most accurate where the density has largest mass.

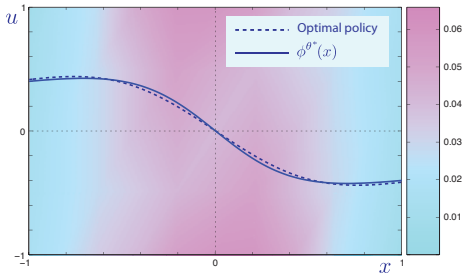

Fig. 2: The same experiment used to obtain Fig. 1 was performed, except that the control input amplitude was increased. This resulted in a stationary distribution ϖ with broader support. It is evident that the approximating policy more nearly matches the optimal policy near $|x| \sim 1$ in this case.

On substituting $u = \phi^*(x)$ into the HJB equation we obtain $\frac{1}{2}x^2 - \frac{1}{2}(\nabla J^*(x))^2 - x^3 \nabla J^*(x) = \gamma J^*(x)$. In the special case $\gamma = 0$ this can be solved:

$$\nabla J^*(x) = -x^3 + \sqrt{x^6 + x^2}$$

For non-zero γ we don't have an explicit solution to the HJB equation, but the order of growth can be estimated to give,

$$\nabla J^*(x) \approx \begin{cases} O(x) & x \sim 0 \\ \frac{1}{2x} + o(\frac{1}{x}) & x \sim \infty \end{cases}$$

The Q-function is equal to the sum $H^*(x,u) = c(x,u) + \nabla J^*(x)(-x^3 + u)$. The following two dimensional parameterization is consistent with these asymptotic expressions:

$$H^\theta(x,u) = c(x,u) + \theta^x x^2 + \theta^{xu} \frac{x}{1+2x^2} u.$$

The SDQ(β)-learning algorithm was run with $\beta = \gamma = 0.1$ and the input

$$u(t) = A(\sin(t) + \sin(\pi t) + \sin(et)),$$

where A is the amplitude of the input. Input trajectories with two amplitudes were applied: $A = 0.2$ and $A = 1$. In either case, the parameters θ^x, θ^{xu} converge to a steady-state value. For $A = 0.2$, Fig. 1 compares the optimal policy

$$\phi^{\theta^*}(x) = -\theta^{xu} \frac{x}{1+2x^2},$$

evaluated at the steady state value of parameters with $\phi^*(x)$, the analytically obtained optimal policy at $\gamma = 0$. Fig. 2 depicts the same comparison for $A = 1$. The figures show that the approximation is consistent on the support of the stationary distribution ϖ.

B. Linear systems

When (22) is linear and $c(x,u) = \frac{1}{2}x^T Q x + \frac{1}{2}u^T R u$ quadratic, then we take the parameterization

$$H^\theta(x,u) = c(x,u) + \tfrac{1}{2}\sum_{i=1}^{d_x} \theta_i^x x^T E^i x + \sum_{j=1}^{d_{xu}} \theta_j^x x^T F^i u \quad (42)$$

where the matrices $\{E^i, F^i\}$ are pre-specified. We also denote the Q-function using the following matrix adaptation of (24),

$$H^\theta(x,u) = c(x,u) + x^T E^\theta x + x^T F^\theta u \quad (43)$$

The minimum (26) becomes

$$\underline{H}^\theta(x) = \tfrac{1}{2}x^T \left(Q + E^\theta - F^{\theta^T} R^{-1} F^\theta \right) x.$$

The policy (11) is given by the linear state feedback law,

$$\phi^\theta(x) = -R^{-1} F^\theta x \quad (44)$$

In the remainder of this example, we summarize the results of the SDQ(γ)-learning for a two dimensional state space model with a single input,

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u,$$

$Q = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, $R = 1$ and $\gamma = 0.1$. The Q-function in (43) has the parametrization:

$$E^\theta = \begin{pmatrix} \theta_1^x & \theta_3^x \\ \theta_3^x & \theta_2^x \end{pmatrix}, \quad F^\theta = \begin{pmatrix} \theta_1^{xu} \\ \theta_2^{xu} \end{pmatrix}.$$

Fig. 3 shows results from SDQ(γ)-learning for these parameters. Two input trajectories were applied: The first was a sum of sinusoids with irrationally related frequencies, and the second equal to the sum of the first and a (scaled) pulse train. These inputs are illustrated on the left. The introduction of impulses speeded convergence of parameters significantly. The Bellman error converged to zero *very quickly* using either input.

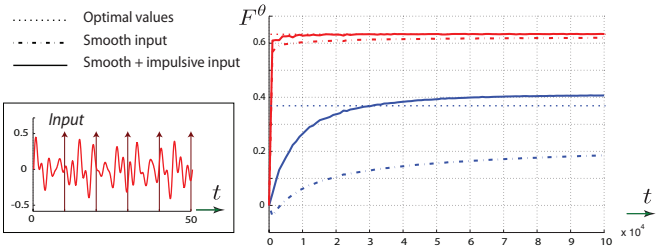

Fig. 3: Sample paths of estimates of the matrix F^* for a linear model.

C. Distributed control of multi-agent systems

Q-learning is a natural candidate for applications in distributed control. We illustrate this with results obtained for the *large-population cost-coupled LQG problem* introduced in [7].

The model consists of n non-homogeneous autonomous agents. The i^{th}-agent is modeled as a scalar linear system

$$\tfrac{d}{dt}x_i = a_i x_i + b_i u_i, \quad (45)$$

where x_i and u_i denote the state and the control of the i^{th}-agent, respectively. The agents are coupled through their respective quadratic cost functions,

$$c_i(x_i, u_i) = (x_i - z)^2 + u_i^2, \quad 1 \le i \le n,$$

where z is the mean, $z = n^{-1}(x_1 + \cdots + x_n)$. For the discounted cost LQ problem, the authors introduce a state aggregation procedure whereby each agent solves the optimal control problem using its own state and the average state of all agents, referred to as the *mass* (see Eq. (4.6)-(4.9) in [7]).

The formulation in [7] suggests that each agent can learn an approximately optimal policy using Q-learning. The "state" of the i^{th}-agent is taken to be $[x_i, z]$ and the Q-function for the LQ problem is defined according to the matrix parametrization (43). As with the previous example, each agent has three parameters (E^θ) that are coefficients of the basis functions $\{x_i^2, z^2, x_i z\}$, and two parameters (F^θ) that are coefficients of the basis functions $\{x_i u_i, z u_i\}$.

We carried out numerical simulations with five agents described by (45) with

$$\begin{bmatrix} \{a_i\} \\ \{b_i\} \end{bmatrix} = \begin{bmatrix} -0.1 & -0.09 & -0.03 & -0.10 & -0.09 \\ 1.0 & 0.66 & 1.11 & -0.75 & 0.013 \end{bmatrix}$$

Apart from the first agent, the parameters for the other agents were picked randomly.

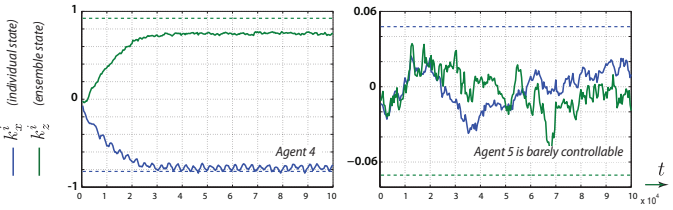

Fig. 4: Sample paths of estimates of (k_x^i, k_z^i) for $i = 4$ and 5, using the SDQ-learning algorithm. The dashed lines show the asymptotically optimal values obtained in [7].

Five applications of the Q-learning algorithm were run in parallel. Each of the five inputs u_i were taken to be sinusoidal, with irrationally related frequencies. Fig. 4 depicts the evolution of estimates of the two components of the local optimal gain (44) for two of the five agents ($i = 4$ and 5), expressed $u_i = -k_x^i x_i - k_z^i z$. Also shown are the gains introduced in [7] that were found to be asymptotically optimal for large n. For all but one of the five agents, the limiting values of the estimates of (k_x^i, k_z^i) were close to those predicted in [7]. The first plot shows typical behavior of the algorithm.

In the sole case where the results appear inconsistent, the magnitude of the optimal control gain is nearly zero. This is because agent 5 is "nearly uncontrollable" with $b_5 = 0.013$.

V. Conclusions

The reader might now ask, should Watkin's algorithm be called H-learning? Or, perhaps D-Q learning? (recall that the Q-function first appeared in the differential dynamic programming framework of Jacobson and Mayne [8]). We leave this decision to the author of the algorithm.

There are many avenues open for future research. We list just a few here:

(i) The algorithm can be refined in many ways. State weighting can be introduced as in LP approaches [5], or TD learning [10]. Variance reduction techniques might be employed for Markovian models [10].

(ii) The distributed control may be considered by selecting suitable basis. Consider for example the basis in Sec. IV-C based on structure for a limiting model with an infinite number of agents (see also [12], [10]).

(iii) Finite dimensional parameterizations of Q-learning invite extensions to the POMDP models and the output feedback case for deterministic models.

Acknowledgment Financial support from the National Science Foundation (ECS-0523620 and CMS 05-56352) and ITMANET DARPA RK 2006-07284 is gratefully acknowledged. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF or DARPA.

References

[1] D.P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Atena Scientific, Cambridge, Mass, 1996.

[2] V. S. Borkar and S. P. Meyn. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control Optim.*, 38(2):447–469, 2000.

[3] S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Mach. Learn.*, 22(1-3):33–57, 1996.

[4] S.J. Bradtke, B.E. Ydstie, and A.G. Barto. Adaptive linear quadratic control using policy iteration. In *Proc. of the 1994 American Control Conference*, volume 3, pages 3475–3479, 1994.

[5] D. P. Pucci de Farias and B. Van Roy. A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees. *Math. Oper. Res.*, 31(3):597–620, 2006.

[6] J. Han and B. Van Roy. Control of diffusions via linear programming. To appear in a volume on stochastic programming in honor of George Dantzig, edited by Gerd Infanger. Preprint available at http://www.stanford.edu/˜bvr/, 2009.

[7] M. Huang, P. E. Caines, and R. P. Malhame. Large-population cost-coupled LQG problems with nonuniform agents: Individual-mass behavior and decentralized ε-Nash equilibria. *IEEE Trans. Auto. Control*, 52(9):1560–1571, 2007.

[8] D. H. Jacobson and D. Q. Mayne. *Differential dynamic programming*. American Elsevier Pub. Co., New York, NY, 1970.

[9] F. S. Melo, S. Meyn, and M. Isabel Ribeiro. An analysis of reinforcement learning with function approximation. *Proc. of ICML*, pages 664–671, 2008.

[10] S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, Cambridge, 2007.

[11] S. P. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Springer-Verlag, London, 1993. 2nd Edition to appear, CUP 2009.

[12] C.C. Moallemi, S. Kumar, and B. Van Roy. Approximate and data-driven dynamic programming for queueing networks, 2008.

[13] Y. Tassa and T. Erez. Least squares solutions of the HJB equation with neural network value-function approximators. *IEEE Trans. on Neural Networks*, 18(4):1031–1041, 2007.

[14] J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Auto. Control*, 42(5):674–690, 1997.

[15] D. Vrabie, M. Abu-Khalaf, F.L. Lewis, and Y. Wang. Continuous-time ADP for linear systems with partially unknown dynamics. In *Proc. IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 247–253, April 2007.

[16] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F.L. Lewis. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica*, 45(2):477 – 484, 2009.

[17] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.