2021 American Control Conference (ACC)
New Orleans, USA, May 25-28, 2021

Convex Q-Learning

Fan Lu, Prashant G. Mehta, Sean P. Meyn and Gergely Neu

***Abstract*— It is well known that the extension of Watkins' algorithm to general function approximation settings is challenging: does the "projected Bellman equation" have a solution? If so, is the solution useful in the sense of generating a good policy? And, if the preceding questions are answered in the affirmative, is the algorithm consistent? These questions are unanswered even in the special case of Q-function approximations that are linear in the parameter. The challenge seems paradoxical, given the long history of convex analytic approaches to dynamic programming.**

Our main contributions are summarized as follows:

(i) A new class of convex Q-learning algorithms is introduced based on a convex relaxation of the Bellman equation. Convergence is established under general conditions for linear function approximation.

(ii) A batch implementation appears similar to LSPI and DQN algorithms, but the difference is substantial: while convex Q-learning solves a convex program that approximates the Bellman equation, theory for DQN is no stronger than for Watkins algorithm with function approximation.

These results are obtained for deterministic nonlinear systems with total cost criterion. Extensions are proposed.

I. Introduction

This paper concerns design of reinforcement learning algorithms for nonlinear, deterministic state space models. The setting is deterministic systems in discrete time, where the main ideas are most easily described. Specifically, we consider a nonlinear state space model

$$x_{k+1} = \mathrm{F}(x_k, u_k)\,, \qquad k \ge 0\,,\ x_0 \in \mathsf{X}\,, \tag{1}$$

with state space X, input (or action) space U, and where $\mathrm{F}\colon \mathsf{X} \times \mathsf{U} \to \mathsf{X}$. (the sets X and U are general).

It is assumed that there is $(x^e, u^e) \in \mathsf{X} \times \mathsf{U}$ that achieves equilibrium:

$$x^e = \mathrm{F}(x^e, u^e).$$

The paper concerns infinite-horizon optimal control, whose definition requires a cost function $c\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}_+$. The cost function is non-negative, and vanishes at (x^e, u^e).

PGM is with the Coordinated Science Laboratory and the Department of Mechanical Science and Engineering at the University of Illinois at Urbana-Champaign (UIUC); SPM and FL are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611; SPM holds an Inria International Chair, Paris, France. Financial support from ARO award W911NF1810334 and National Science Foundation award EPCN 1935389 is gratefully acknowledged; GN is with the Department of Information and Communication Technologies, Universitat Pompeu Fabra (Barcelona, Spain). GN was supported by "la Caixa" Banking Foundation through the Junior Leader Postdoctoral Fellowship Programme, a Google Faculty Research Award, and a Bosch AI Young Researcher Award. This work was done in part while SPM and GN were participating in a program at the Simons Institute for the Theory of Computing.

The (optimal) value function is denoted

$$J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c(x_k, u_k)\,, \quad x_0 = x \in \mathsf{X}\,, \tag{2}$$

where the minimum is over all input sequences $\boldsymbol{u} := \{u_k : k \ge 0\}$. The goal of optimal control is to find an optimizing input sequence, and in the process we often need to compute the value function $J^\star$. We settle for an approximation in the majority of cases. In this paper the approximations will be based on Q-learning; it is hoped that the main ideas will be useful in other formulations of reinforcement learning.

Background The "Q-function" of reinforcement learning is

$$Q^\star(x,u) := c(x,u) + J^\star(\mathrm{F}(x,u)), \tag{3}$$

and the usual Bellman equation is expressed as

$$J^\star(x) = \min_{u \in \mathsf{U}(x)} Q^\star(x,u), \tag{4}$$

where $\mathsf{U}(x) \subset \mathsf{U}$ captures input-constraints. This implies the following fixed-point equation for the Q-function:

$$Q^\star(x,u) = c(x,u) + \underline{Q}^\star(\mathrm{F}(x,u)), \tag{5}$$

with $\underline{Q}(x) = \min_u Q(x,u)$ for any function Q. The optimal input is state feedback $u_k^\star = \phi^\star(x_k^\star)$, with

$$\phi^\star(x) \in \arg\min_u Q^\star(x,u)\,, \qquad x \in \mathsf{X}. \tag{6}$$

Temporal difference (TD) and Q-learning are two large families of reinforcement learning algorithms based on approximating the value function or Q-function as a means to approximate $\phi^\star$ [40], [6], [5]. Consider a parameterized family $\{Q^\theta : \theta \in \mathbb{R}^d\}$, each a real-valued function on $\mathsf{X} \times \mathsf{U}$. For example, the vector θ might represent weights in a neural network. Q-learning algorithms are designed to approximate $Q^\star$ within this parameterized family. Given the parameter estimate $\theta \in \mathbb{R}^d$, the "Q^θ-greedy policy" is obtained:

$$\phi^\theta(x) = \arg\min_u Q^\theta(x,u) \tag{7}$$

Ideally, we would like to find an algorithm that finds a value of θ so that it best approximates the optimal policy.

Most algorithms use the following interpretation of (5):

$$Q^\star(x_k,u_k) = c(x_k,u_k) + \underline{Q}^\star(x_{k+1}) \tag{8}$$

valid for any input-state sequence $\{u_k, x_k : k \ge 0\}$. Two general approaches to define θ^* are each posed in terms of the *temporal difference*:

$$\mathcal{D}_{k+1}(\theta) := -Q^\theta(x_k,u_k) + c(x_k,u_k) + \underline{Q}^\theta(x_{k+1}) \tag{9}$$

978-1-6654-4197-1/$31.00 ©2021 AACC

assumed to be observed on the time-horizon $0 \le k \le N$.

(i) Often considered a *gold standard* loss function is the mean-square Bellman error associated with (9):

$$\mathcal{E}^{\varepsilon}(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}\big[\mathcal{D}_{k+1}(\theta)\big]^2 \tag{10}$$

(ii) Watkins' Q-learning algorithm, as well as the earlier TD methods of Sutton can be cast as a *Galerkin relaxation* of the Bellman equation: A sequence of d-dimensional *eligibility vectors* $\{\zeta_k\}$ is constructed, and the goal then is to solve

$$0 = \frac{1}{N}\sum_{k=0}^{N-1}\mathcal{D}_{k+1}(\theta)\zeta_k \tag{11}$$

In the original algorithm of Watkins,

▲ The algorithm is defined for finite state and action MDPs (Markov Decision Processes), and $Q^\star \in \{Q^\theta : \theta \in \mathbb{R}^d\}$ (the goal is to compute the Q-function exactly).

▲ The approximation family is linear $Q^\theta = \theta^\intercal\psi$, with $\psi\colon \mathsf{X}\times\mathsf{U}\to\mathbb{R}^d$, and $\zeta_k = \psi(x_k, u_k)$.

Equation (11) is not how Q-learning algorithms are typically presented, but approximates the goal in many formulations. A limit point of Watkins' algorithm, and generalizations such as [28], [23], solves the "projected Bellman equation":

$$\overline{f}(\theta^*) = 0\,, \qquad \overline{f}(\theta) = \mathsf{E}\big[\mathcal{D}_{k+1}(\theta)\zeta_k\big] \tag{12}$$

where the expectation is in steady-state (one assumption being the existence of a steady-state). The basic extension of Watkins' algorithm is defined by the recursion

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n \tag{13}$$

with $\{\alpha_n\}$ the non-negative step-size sequence, and $\mathcal{D}_{n+1}$ is short-hand for $\mathcal{D}_{n+1}(\theta_n)$. The recursion (13) is called Q(0)-learning for the special case $\zeta_n = \nabla_\theta Q^\theta(x_n,u_n)\big|_{\theta=\theta_n}$, in analogy with TD(0)-learning [16], [29]. Criteria for convergence is typically cast within the theory of stochastic approximation, which is based on the ODE,

$$\tfrac{d}{dt}\vartheta_t = \overline{f}(\vartheta_t)\,, \qquad \vartheta_0 \in \mathbb{R}^d \tag{14}$$

Conditions for convergence are very restrictive [28], [23].

While not obvious from its description, the DQN algorithm (a significant component of famous applications such as AlphaGo) will converge to the same projected Bellman equation, provided it is convergent—see Prop. 3.3 below.

This opens an obvious question: does (12) have a solution? Does the solution lead to a good policy? An answer to the second question is wide open, despite the success in applications. The answer to the first question is, in general, *no*. The conditions imposed in [28] for a solution are very strong, and not easily verified in any applications; the more recent work [23] offers improvements, but nothing approaching a full understanding of the algorithm.

The question of existence led to an entirely new approach in [24], based on the non-convex optimization problem:

$$\min_\theta J(\theta) = \min_\theta \tfrac{1}{2}\overline{f}(\theta)^\intercal M\overline{f}(\theta), \qquad \text{with} \quad M>0 \tag{15}$$

and $\overline{f}$ defined in (12). Consider the continuous-time gradient descent algorithm associated with (15):

$$\tfrac{d}{dt}\vartheta_t = -[\partial_\theta\overline{f}\,(\vartheta_t)]^\intercal M\overline{f}(\vartheta_t) \tag{16}$$

The GQ-learning algorithm is a stochastic approximation (SA) translation of this ODE, using $M = \mathsf{E}[\zeta_n\zeta_n^\intercal]^{-1}$. Convergence holds under conditions, such as a coercive condition on J (which is not easily verified a-priori). Most important: does this algorithm lead to a good approximation of the Q-function, or the policy?

This brings us back to (ii): *why approximate the solution of* (12)*?* We do not have an answer. In this paper we return to (10), for which we seek convex re-formulations.

Contributions. The apparent challenge with approximating the Q-function is that it solves a nonlinear fixed point equation (5) or (8), for which root finding problems for approximation may not be successful (as counter examples show). This challenge seems paradoxical, given the long history of convex analytic approaches to dynamic programming in both the MDP literature [25], [15], [7] and the linear optimal control literature [42].

The starting point of this paper is to clarify this paradox, and from this create a new family of RL algorithms designed to minimize a convex variant of the empirical mean-square error (10). This step was anticipated in [26], for which a convex formulation of Q-learning was proposed for deterministic systems in continuous time.

The main contributions are summarized as follows:

(i) We introduce a variant of the classic linear-programming (LP) formulation of dynamic programming [25], [15], [7], [13], [14] that lends itself to data driven RL algorithms for nonlinear control systems. The key feature of our formulation absent from previous work is the introduction of the Q-function in the LP. The precise formulation of the newly introduced "DPLP" (dynamic programming linear program) is given in Prop. 2.1 for deterministic control systems, with generalizations to MDPs in [27]. The relationship with semi-definite programs for LQR is made precise in Prop. 2.2

(ii) New Q-learning algorithms inspired by the DPLP. A casual reader might mistake some of these algorithms (e.g., the one given in Equation (42)) for the DQN algorithm, although there are several subtle differences that have important implications, as made clear in Propositions 3.1 and 3.3: the DQN algorithm cannot solve the minimal mean-square Bellman error optimization problem. Rather, any limit of the algorithm must solve the relaxation (12) (recall that there is little theory surrounding existence or interpretation of solutions to this fixed point equation).

(iii) Under mild conditions, including a linear function approximation architecture, parameter estimates from the Batch Convex Q algorithm will converge to the solution to the convex program

$$\begin{aligned}\min_\theta \;& -\langle\mu, Q^\theta\rangle + \kappa^{\varepsilon}\mathsf{E}\big[\big\{\mathcal{D}^{\circ}_{k+1}(\theta)\big\}_-^2\big]\\ \text{s.t.}\;& \mathsf{E}\big[\min_i\{\mathcal{D}^{\circ}_{k+1}(\theta)\zeta^{\varepsilon}_k(i)\}\big]\ge 0\end{aligned} \tag{17}$$

where $\mathcal{D}^{\circ}_{k+1}(\theta)$ denotes the *observed Bellman error*:

$$\mathcal{D}^{\circ}_{k+1}(\theta) := -Q^{\theta}(x_k, u_k) + c(x_k, u_k) + \underline{Q}^{\theta}(x_{k+1})$$

The expectations in (17) are in steady-state. The remaining variables are part of the algorithm design: $\{\zeta^{\varepsilon}_k\}$ is a non-negative vector-valued sequence, μ is a positive measure, and κ^{ε} is non-negative.

The convex program (17) is a relaxation of the DPLP, which is tight under ideal conditions (including the assumption that $Q^{\star}$ is contained in the function class). See Corollary 3.2 for details.

As in [26], it is argued that there is no reason to introduce random noise for exploration for RL in deterministic control applications. Rather, we opt for the quasi-stochastic approximation (QSA) approach of [26], [11].

Literature review This paper began as a resurrection of the conference paper [26], which deals with Q-learning for deterministic systems in continuous time. A primitive version of convex Q-learning was introduced; the challenge at the time was to find ways to create a reliable online algorithm. This is resolved in the present paper through a combination of Galerkin relaxation techniques and ERM.

Two recent related papers appeared in an MDP setting: [22] is based on a variant of the convex program (21), with the inequality in (21b) replaced by equality, and [33] derives optimistic exploration algorithms for episodic MDPs from an LP formulation closely related to the DPLP presented here. Neither of these authors were aware of the prior 2009 work [26], in which (21) was first introduced in continuous time. The article [22] contains substantial theory for a very special case: the tabular MDP setting in which the function class parameterizes all possible Q-functions. Further theoretical results for linear function approximation are presented in Sections 2 and 5 of [33]. We see no overlap in contributions.

The ideas in [26] and the present paper were inspired in part by the LP approach to dynamic programming introduced by Manne in the 60's [25], [15], [1], [7]. A significant program on linear programming approaches to approximate dynamic programming is presented in [13], [14], but we do not see much overlap with the present work. This line of research finds its roots in the early work of Schweitzer and Seidmann [39], which is coincidentally the first reference that we are aware of to propose the loss function (10) for approximate dynamic programming. There is a also on-going research on LP approaches to optimal control for deterministic systems [43], [20], [9], and semi-definite programs (SDPs) in linear optimal control [44].

Deep Q-Learning (DQN) algorithms have been first proposed by Riedmiller [37] and have started to attract attention following their breakthrough application to train human-level Atari agents by Mnih et al. [31], [32]. Since then, numerous subsequent improvements have been made to the basic DQN method [41], [45], [3], [12], eventually culminating in the "Rainbow DQN" algorithm of Hessel et al. [19]. Throughout this subsequent work, the squared Bellman error objective (10) has remained an important core component of DQN algorithms (although it has been sometimes replaced by even more complicated loss functions depending on the entire distribution of the squared errors [3], [12]), and it has been used extensively in other reinforcement learning algorithms that are not based on approximate dynamic programming [38], [30], [18], [17]. We do not know if the research community is aware that DQN algorithms, if convergent, will converge to the projected Bellman equation (12), and that an ODE approximation of DQN is identical to that of (13) (for which stability theory is currently very weak). A formal definition of "ODE approximation" is provided at the end of Section II.

The concurrent work of Bas-Serrano et al. [2] bears several similarities to the present paper: most notably, they present another convex alternative to the squared Bellman error (10) called the logistic Bellman error, which is also derived from the same DPLP we use as our starting point. That said, the two papers diverge significantly when it comes to the details of the algorithm design and the nature of the performance guarantees. Unifying the advantages of the two approaches is an exciting direction for future research.

The remainder of the paper is organized as follows. Section II begins with a review of linear programs for optimal control, and new formulations designed for application in RL. Section III shows how concepts from least squares value iteration can be adopted to obtain efficient approximations of the DPLP for application in RL. Their relationship with DQN (as well as significant advantages) is explained. The theory is illustrated with a single example in Section IV. Section V contains conclusions and topics for future research. Technical proofs are contained in [27].

II. Convex Q-Learning

The RL algorithms introduced in this paper are designed to approximate the value function $J^{\star}$ within a finite-dimensional function class $\{J^{\theta} : \theta \in \mathbb{R}^d\}$. We obtain a convex program when this parameterization is linear.

As surveyed in the introduction, a favored approach in Q-learning is to consider for each $\theta \in \mathbb{R}^d$ and $x \in \mathsf{X}$, the associated Bellman error

$$\mathcal{B}^{\theta}(x) = -J^{\theta}(x) + \min_{u}[c(x,u) + J^{\theta}(\mathrm{F}(x,u))] \tag{18}$$

A generalization of (10) is to introduce a weighting measure μ (typically a probability mass function (pmf) on X), and consider the mean-square Bellman error

$$\mathcal{E}(\theta) = \int [\mathcal{B}^{\theta}(x)]^2 \, \mu(dx) \tag{19}$$

A significant challenge is that the loss function $\mathcal{E}$ is not convex. We obtain a convex optimization problem that is suitable for application in RL by approximating simultaneously $J^{\star}$ and $Q^{\star}$, where the Q-function is defined in (3).

Proofs of all technical results can be found in [27].

Bellman Equation is a Linear Program For any $J\colon \mathsf{X} \to \mathbb{R}$, and any $r \in \mathbb{R}$, let $S_J(r)$ denote the *sub-level set*:

$$S_J(r) = \{x \in \mathsf{X} : J(x) \le r\} \tag{20}$$

The function J is called *inf-compact* if the set $S_J(r)$ is either pre-compact, empty, or $S_J(r) = \mathsf{X}$ (depending on r).

Proposition 2.1: Suppose that the value function $J^\star$ defined in (2) is continuous, inf-compact, and vanishes only at x^e. Then, for any positive measure μ on $\mathsf{X} \times \mathsf{U}$, the pair $(J^\star, Q^\star)$ solve the following convex program:

$$\max_{J,Q} \ \langle \mu, Q \rangle \tag{21a}$$

$$\text{s.t.} \quad Q(x,u) \le c(x,u) + J(\mathsf{F}(x,u)) \tag{21b}$$

$$Q(x,u) \ge J(x)\,, \qquad x \in \mathsf{X},\ u \in \mathsf{U}(x) \tag{21c}$$

$$J \text{ is continuous, and } J(x^e) = 0. \tag{21d}$$

The LP (21) is more complex than found in the literature [1], but is far more easily adapted to RL applications. To see why, define for any (J, Q) the Bellman error:

$$\mathcal{D}^\circ(J,Q)_{(x,u)} := -Q(x,u) + c(x,u) + J(\mathsf{F}(x,u)) \tag{22}$$

Similar to the term $\mathcal{D}_{n+1}$ appearing in (13), we have for any input-state sequence,

$$\mathcal{D}^\circ(J,Q)_{(x_k,u_k)} = -Q(x_k,u_k) + c(x_k,u_k) + J(x_{k+1})$$

Hence we can observe the Bellman error along the sample path. The right hand side will be called the temporal difference, generalizing the standard terminology.

Semi-definite program for LQR Consider the LTI model:

$$x_{k+1} = Fx_k + Gu_k, \qquad x_0 = x \tag{23a}$$

$$y(k) = Hx_k \tag{23b}$$

where (F, G, H) are matrices of suitable dimension (in particular, F is $n \times n$ for an n-dimensional state space), and assume that the cost is quadratic:

$$c(x,u) = \|y\|^2 + u^\intercal R u \tag{24}$$

with $y = Hx$, and $R > 0$. We denote $S = H^\intercal H \ge 0$.

The variables in the LP (21) can be restricted to quadratics in this special case, and with this restriction it is equivalent to a semi-definite program:

Proposition 2.2: Suppose that (F, G) is stabilizable and (F, H) is detectable, so that $J^\star$ is everywhere finite. Then, the value function and Q-function are each quadratic: $J^\star(x) = x^\intercal M^\star x$ for each x, where $M^\star \ge 0$ is a solution to the algebraic Riccati equation. The matrix $M^\star$ is also the solution to the following convex program:

$$M^\star \in \arg\max \ \big(\text{trace}\,(M)\big)$$
$$\text{s.t.} \quad \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} F^T M F & F^T M G \\ G^T M F & G^T M G \end{bmatrix} \ge \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$$

where the maximum is over symmetric matrices M, and the inequality constraint is in the sense of symmetric matrices.

LQR and DQN For the linear system (23) with quadratic cost (24), the Q-function is a quadratic. To apply Q-learning we might formulate a linear parameterization:

$$Q^\theta(x,u) = x^\intercal M_F x + 2u^\intercal N x + u^\intercal M_G u$$

in which the three matrices depend linearly on θ. The Q-learning algorithm (13) and the DQN algorithm require the minimum of the Q-function, which is easily obtained:

$$\underline{Q}^\theta(x) = \min_u Q^\theta(x,u) = x^\intercal \big\{M_F - N^\intercal M_G^{-1} N\big\} x$$

Consequently, the function $\underline{Q}^\theta$, and $\overline{f}$ defined in (12), are unlikely to be Lipschitz continuous.

Exploration and Quasi-Stochastic Approximation The success of an RL algorithm based on temporal difference methods depends on the choice of input $\boldsymbol{u}$ during training. The purpose of this section is to make this precise, and present our main assumption on the input designed for generating data to train the algorithm (also known as *exploration*).

Throughout the paper it is assumed that the input used for training is state-feedback with perturbation:

$$u_k = \phi(x_k, \xi_k) \tag{25}$$

where the exploration signal $\boldsymbol{\xi}$ is a bounded sequence evolving on $\mathbb{R}^p$ for some $p \ge 1$. As a means to reduce variance we adopt the quasi-stochastic approximation (QSA) setting of [26], [4]. For example, ξ_k may be a mixture of sinusoids of irrational frequencies.

It will be convenient to assume that the exploration is Markovian, which for a deterministic sequence means it is the state process for a nonlinear state space model:

$$\xi_{k+1} = \mathrm{H}(\xi_k) \tag{26}$$

It is assumed that $\mathrm{H}\colon \mathbb{R}^p \to \mathbb{R}^p$ is continuous. Subject to the policy (25), it follows that the triple $\Phi_k = (x_k, u_k, \xi_k)^\intercal$ is also Markovian, with state space $\mathsf{Z} = \mathsf{X} \times \mathsf{U} \times \mathbb{R}^p$. The continuity assumption imposed below ensures that $\boldsymbol{\Phi}$ has the Feller property, and boundedness of $\boldsymbol{\Phi}$ from just one initial condition implies the existence of an invariant probability measure that defines the "steady state" behavior [29].

Denote, for any continuous function $g\colon \mathsf{Z} \to \mathbb{R}$,

$$\overline{g}_N = \frac{1}{N} \sum_{k=1}^{N} g(\Phi_k)\,, \qquad N \ge 1$$

For any $L > 0$ denote

$$\mathcal{G}_L = \{g : \|g(z') - g(z)\| \le L\|z - z'\|, \ \text{for all} \ \ z, z' \in \mathsf{Z}\}$$

The following is assumed throughout this section.

(A$\boldsymbol{\xi}$) The state and action spaces X and U are Polish spaces; F defined in (1), ϕ defined in (25), and H in (26) are each continuous on their domains, and the larger state process $\boldsymbol{\Phi}$ is bounded and ergodic in the following sense: There is a unique probability measure ϖ, with compact support, such that for any continuous function $g\colon \mathsf{Z} \to \mathbb{R}$,

the following ergodic average exists for each initial condition Φ_0

$$\mathsf{E}_\varpi[g(\Phi)] := \lim_{N\to\infty} \overline{g}_N \tag{27}$$

Moreover, the limit is uniform: for each $L < \infty$,

$$\lim_{N\to\infty} \sup_{g\in\mathcal{G}_L} |\overline{g}_N - \mathsf{E}_\varpi[g(\Phi)]| = 0 \qquad \square$$

For analysis of Q-learning with function approximation, the vector field introduced in (12) is defined similarly: $\overline{f}(\theta) =$

$$\lim_{N\to\infty} \frac{1}{N}\sum_{k=1}^{N}\Big[-Q^\theta(x_k,u_k) + c(x_k,u_k) + \underline{Q}^\theta(x_{k+1})\Big]\zeta_k^\varepsilon$$

The choice of "exploration policy" (25) is imposed mainly to simplify analysis. We might speed convergence significantly with an "epsilon-greedy policy":

$$u_k = \phi^{\theta_k}(x_k, \xi_k)$$

in which the right hand side is a perturbation of the exact Q^θ-greedy policy (7), using current estimate θ_k. There is a growing literature on much better exploration schemes, motivated by techniques in the bandits literature [34], [35]. The marriage of these techniques with the algorithms introduced in this paper is a subject for future research.

ODE approximations The technical results that follow require that we make precise what we mean by *ODE approximation*. Consider a recursion of the form

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f_{n+1}(\theta_n)\,, \qquad n \ge 0 \tag{28}$$

in which $\{f_n\}$ admits an ergodic limit:

$$\overline{f}(\theta) := \lim_{N\to\infty} \frac{1}{N}\sum_{k=1}^{N} f_k(\theta)\,, \qquad \theta \in \mathbb{R}^d$$

The associated ODE is defined using this vector field:

$$\tfrac{d}{dt}\vartheta_t = \overline{f}(\vartheta_t) \tag{29}$$

An ODE approximation is defined by mimicking the usual Euler construction: the time-scale for the ODE is defined by the non-decreasing time points $t_0 = 0$ and $t_n = \sum_0^n \alpha_k$ for $n \ge 1$. Define a continuous time process $\boldsymbol{\Theta}$ by $\Theta_{t_n} = \theta_n$ for each n, and extend to all t through piecewise linear interpolation. The next step is to fix a time horizon for analysis of length $\mathcal{T} > 0$ (its choice is determined based on properties of the ODE). Denote $\mathcal{T}_0 = 0$, and

$$\mathcal{T}_{n+1} = \min\{t_n : t_n - \mathcal{T}_n \ge \mathcal{T}\}\,, \qquad n \ge 0 \tag{30}$$

Let $\{\vartheta_t^n : t \ge \mathcal{T}_n\}$ denote the solution to the ODE (29) with initial condition $\vartheta^n_{\mathcal{T}_n} = \theta_{k(n)}$, with index defined so that $t_{k(n)} = \mathcal{T}_n$. We then say that the algorithm (28) admits an *ODE approximation* if for each initial θ_0,

$$\lim_{n\to\infty} \sup_{\mathcal{T}_n \le t \le \mathcal{T}_{n+1}} \|\Theta_t - \vartheta_t^n\| = 0 \tag{31}$$

Basic algorithm The RL algorithms introduced in this paper are all motivated by the "DPLP" (21). We search for an approximate solution among a finite-dimensional family $\{J^\theta\,,\ Q^\theta : \theta \in \mathbb{R}^d\}$. The value θ_i might represent the ith weight in a neural network function approximation architecture, but to justify the adjective *convex* we require a linear family:

$$J^\theta(x) = \theta^\intercal \psi^J(x)\,, \qquad Q^\theta(x,u) = \theta^\intercal \psi(x,u) \tag{32}$$

normalized with $J^\theta(x^e) = 0$ for each θ.

The temporal difference sequence is a modification of (9):

$$\mathcal{D}^\circ_{k+1}(\theta) := -Q^\theta(x_k,u_k) + c(x_k,u_k) + J^\theta(x_{k+1}) \tag{33}$$

The algorithms are designed so that J^θ approximates $\underline{Q}^\theta$, and hence $\mathcal{D}^\circ_{k+1}(\theta) \approx \mathcal{D}_{k+1}(\theta)$ (recall from below (5), $\underline{Q}(x) = \min_u Q(x,u)$ for any function Q). For this it is useful to introduce an additional loss function:

$$\mathcal{E}^\varepsilon(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}\big[\mathcal{D}^\circ_{k+1}(\theta)\big]^2 \tag{34}$$

The first of several versions of "CQL" involves a Galerkin relaxation of the constraints in the DPLP (21). This requires specification of two vector valued sequences $\{\zeta_k^\varepsilon, \zeta_k^+\}$ based on the data, and denote

$$z^\varepsilon(\theta) = \frac{1}{N}\sum_{k=0}^{N-1} \min_i\big\{\mathcal{D}^\circ_{k+1}(\theta)\zeta_k^\varepsilon(i)\big\} \tag{35a}$$

$$z^+(\theta) = \frac{1}{N}\sum_{k=0}^{N-1} \max_i\big\{\big[J^\theta(x_k) - Q^\theta(x_k,u_k)\big]\zeta_k^+(i)\big\} \tag{35b}$$

with temporal difference sequence $\{\mathcal{D}^\circ_{k+1}(\theta)\}$ defined in (33). It is assumed that the vector valued sequences $\{\zeta_k^\varepsilon, \zeta_k^+\}$ take values in $\mathbb{R}_+^M$ for some $M > 1$.

In addition we introduce the convex loss functions:

$$\mathcal{E}^+(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}\big[\{J^\theta(x_k) - Q^\theta(x_k,u_k)\}_+\big]^2 \tag{36a}$$

$$\textit{or}\quad \mathcal{E}^+(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}\big[\{J^\theta(x_k) - \underline{Q}^\theta(x_k)\}_+\big]^2 \tag{36b}$$

where $\{q\}_+ = \max(q,0)$. The second option (36b) more strongly penalizes deviation from the constraint $\underline{Q}^\theta \ge J^\theta$. The choice of definition (36a) or (36b) will depend on the relative complexity, which is application-specific.

Convex Q-Learning For positive scalars κ^ε and κ^+, and a tolerance $\text{Tol} \ge 0$,

$$\theta^* = \arg\min_\theta \big\{-\langle\mu, Q^\theta\rangle + \kappa^\varepsilon\mathcal{E}^\varepsilon(\theta) + \kappa^+\mathcal{E}^+(\theta)\big\} \tag{37a}$$

$$\text{s.t.}\quad z^\varepsilon(\theta) \ge -\text{Tol} \tag{37b}$$

$$z^+(\theta) \le \text{Tol} \tag{37c}$$

The following result shows that care must be taken in choice of μ. The proof is straightforward.

Proposition 2.3: Consider the tabular setting of Watkins: both input space and state space are finite, $Q^\theta = \theta^\intercal\psi$, and ψ_i is the indicator function of pair (x^i, u^i). Suppose there is a pair $(x^\bullet, u^\bullet)$ that is never visited. Then the value of the convex program (37) is infinite. $\square$

III. Batch Convex Q-Learning

To understand why the optimization problem (37) may present challenges, consider their implementation based on a kernel. Either of these optimization problems is a convex program. However, due to the Representer Theorem, the dimension of θ is equal to the number of observations N. Even in simple examples, the value of N for a reliable estimate may be larger than one million. Another challenge observed in numerical experiments is the sensitivity to hard constraints, in particular the constraint $Q^\theta(x,u) \geq J^\theta(x)$ for all x, u. The batch RL algorithms described here are designed to reduce complexity, and there are many other potential benefits [21].

BCQL We proceed in two steps. First, we remove the function J from the DPLP (21). On replacing J with $\underline{Q}$ we obtain the convex program:

$$\max_{Q} \quad \langle \mu, \underline{Q} \rangle \tag{38a}$$

$$\text{s.t.} \quad Q(x,u) \leq c(x,u) + \underline{Q}(\mathsf{F}(x,u)) \,, \ \textit{all } x,u \tag{38b}$$

$$Q \text{ is continuous, and } \underline{Q}(x^e) = 0. \tag{38c}$$

The objective and constraints are convex because $\underline{Q}$ is a concave functional of Q.

It may appear that we have introduced complexity, but this is easily resolved. Consider first the objective: The proof of Prop. 2.1 can be modified to establish $\underline{Q} \leq J^\star$ for any feasible Q. This and (38b) imply

$$Q^\star(x,u) := c(x,u) + J^\star(\mathsf{F}(x,u)) \geq Q(x,u)$$

So, we can simplify (38a) to

$$\max_{Q} \quad \langle \mu, Q \rangle \tag{39}$$

with μ now a probability measure on $\mathsf{X} \times \mathsf{U}$.

The next question is how to reduce the complexity of the constraint (38b). For this, we apply batch RL methods [21]. We first present two algorithms for which we currently have strong theoretical results.

The time-horizon N is broken into B batches of more reasonable size, defined by the sequence of intermediate times $T_0 = 0 < T_1 < T_2 < \cdots < T_{B-1} < T_B = N$, and for $0 \leq n \leq B-1$ we introduce the loss function

$$\mathcal{E}_n^\varepsilon(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[\{-\mathcal{D}_{k+1}^\circ(\theta)\}_+\big]^2 \tag{40a}$$

$$\mathcal{D}_{k+1}^\circ(\theta) := -Q^\theta(x_k,u_k) + c(x_k,u_k) + \underline{Q}^\theta(x_{k+1}) \tag{40b}$$

with $r_n = 1/(T_{n+1} - T_n)$. The introduction of the positive part in (40a) is required to preserve convexity. Also required are a sequence of *regularizers*: $\mathcal{R}_n(Q,\theta)$ is a convex functional of Q, θ, that may depend on θ_n.

Batch Convex Q-Learning 1 With $\theta_0 \in \mathbb{R}^d$ given, along with a sequence of positive scalars $\{\kappa_n^\varepsilon\}$, define

$$\theta_{n+1} = \arg\min_\theta \big\{ -\langle \mu, Q^\theta \rangle + \kappa_n^\varepsilon \mathcal{E}_n^\varepsilon(\theta) + \mathcal{R}_n(Q^\theta, \theta) \big\} \tag{41a}$$

The constraints (37b, 37c) are relaxed in BCQL only to streamline the discussion that follows.

How to choose a regularizer? It is expected that design of $\mathcal{R}_n$ will be inspired by proximal algorithms, so that it will include a term of the form $\|\theta - \theta_n\|^2$ (most likely a weighted norm—see optimal choice in [27]). With a simple scaled norm, the update defining θ_{n+1} becomes

$$\arg\min_\theta \Big\{ -\langle \mu, Q^\theta \rangle + \kappa_n^\varepsilon \mathcal{E}_n^\varepsilon(\theta) + \frac{1}{\alpha_{n+1}} \tfrac{1}{2} \|\theta - \theta_n\|^2 \Big\} \tag{42}$$

where $\{\alpha_n\}$ plays a role similar to a step-size.

Convergence in this special case is established in the following result, based on the steady-state expectation

$$\bar{\mathcal{E}}^\varepsilon(\theta) = \mathsf{E}_\varpi \big[[\{-\mathcal{D}_{k+1}^\circ(\theta)\}_+]^2 \big] \tag{43}$$

Proposition 3.1: Consider the BCQL algorithm (42) subject to the following assumptions:

(i) The parameterization $\{Q^\theta\}$ is linear, and $\bar{\mathcal{E}}^\varepsilon$ is strongly convex, with Lipschitz gradient.

(ii) $\alpha_n = \alpha_1/n$, with $\alpha_1 > 0$.

(iii) The parameters reach steady-state limits:

$$r := \lim_{n\to\infty} r_n \,, \quad \kappa^\varepsilon := \lim_{n\to\infty} \kappa_n^\varepsilon$$

Then, the algorithm is consistent: $\theta_n \to \theta^*$ as $n \to \infty$, where the limit is the unique optimizer:

$$\theta^* = \arg\min_\theta \Big\{ -\langle \mu, Q^\theta \rangle + \kappa^\varepsilon \bar{\mathcal{E}}^\varepsilon(\theta) \Big\} \tag{44}$$

The proof is based on recent SA theory [8], [27].

To justify (17), we must bring back the inequality constraints on the temporal difference, consistent with the DPLP constraint (21b). The batch version of (35a) is denoted

$$z_n^\varepsilon(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \min_i \big\{ \mathcal{D}_{k+1}^\circ(\theta) \zeta_k^\varepsilon(i) \big\} \tag{45}$$

We require $\zeta_k^\varepsilon(i) \geq 0$ for all k, i to ensure that the inequality constraint $z_n^\varepsilon(\theta) \geq 0$ is consistent with (21b). We then arrive at a primal-dual refinement of BCQL1:

Batch Convex Q-Learning 2 With $\theta_0 \in \mathbb{R}^d$, $\lambda_0 \in \mathbb{R}_+$ given, along with a positive scalar κ^ε, consider

$$\begin{aligned} \theta_{n+1} = \arg\min_\theta \Big\{ & -\langle \mu, Q^\theta \rangle + \kappa^\varepsilon \mathcal{E}_n^\varepsilon(\theta) \\ & - \lambda_n z_n^\varepsilon(\theta) + \frac{1}{\alpha_{n+1}} \tfrac{1}{2} \|\theta - \theta_n\|^2 \Big\} \end{aligned} \tag{46a}$$

$$\lambda_{n+1} = \big[\lambda_n - \delta_{n+1} z_n^\varepsilon(\theta_n) \big]_+ \tag{46b}$$

Corollary 3.2: Suppose that assumptions (i) and (ii) of Prop. 3.1 hold. Then, the sequence $\{\theta_n, \lambda_n\}$ obtained from the pd-BCQL algorithm (46) is convergent to a pair (θ^*, λ^*), and the following hold:

(i) The limit θ^* is the solution of the convex program (17), and $\lambda^* \in \mathbb{R}_+$ is the Lagrange multiplier for the inequality constraint in (17).

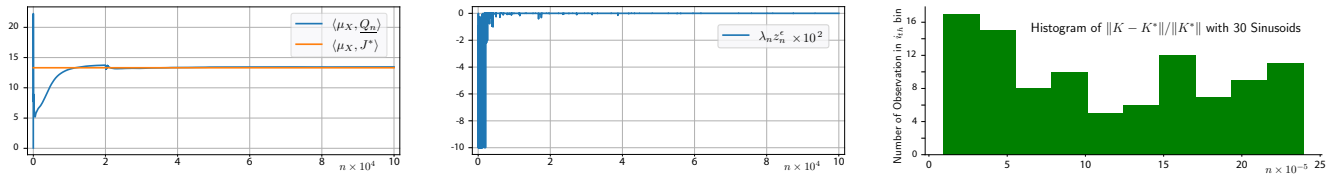

Fig. 1: LQR optimal control using BCQL2

(ii) Consider the special case: the state space and action space are finite, μ has full support, and the dimension of ζ_k^ε is equal to $d_\zeta = |\mathsf{X}| \times |\mathsf{U}|$, with

$$\zeta_k^\varepsilon(i) = \mathbf{1}\{(x_k, u_k) = (x^i, u^i)\}$$

where $\{(x^i, u^i)\}$ is an enumeration of all state action pairs. Suppose moreover that $Q^\star$ is contained in the function class. Then, $Q^\theta\big|_{\theta=\theta^*} = Q^\star$ □

Corollary 3.2 is a corollary to Prop. 3.1, in the sense that it follows the same proof for the joint sequence $\{\theta_n, \lambda_n\}$. Both the proposition and corollary start with a proof that $\{\theta_n\}$ is a bounded sequence, using the "Borkar-Meyn" Theorem [10], [8], [36]. Once boundedness is established, the next step is to show that the algorithm (46) can be approximated by a primal-dual ODE for a saddle point problem.

Comparisons with Deep Q-Learning The *Deep Q Network* (DQN) algorithm was designed for neural network function approximation; the term "deep" refers to a large number of hidden layers. The basic algorithm is summarized below, without imposing any particular form for Q^θ. The definition of $\{T_n\}$ is exactly as in the BCQL algorithm.

DQN With $\theta_0 \in \mathbb{R}^d$ given, along with a sequence of positive scalars $\{\alpha_n\}$, define recursively,

$$\theta_{n+1} = \arg\min_\theta \Big\{ \kappa^\varepsilon \mathcal{E}_n^{\text{DQN}}(\theta) + \frac{1}{\alpha_{n+1}} \|\theta - \theta_n\|^2 \Big\} \quad \text{(47a)}$$

where for each n:

$$\mathcal{E}_n^{\text{DQN}}(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[-\mathcal{D}_{k+1}^n(\theta)\big]^2 \quad \text{(47b)}$$

and $\mathcal{D}_{k+1}^n(\theta) = -Q^\theta(x_k, u_k) + c(x_k, u_k) + Q^{\theta_n}(x_{k+1})$

DQN appears to be nearly identical to (42). Prop. 3.3 (along with Prop. 3.1 and its corollary) show that this resemblance is superficial—*the potential limits of the algorithms are entirely different*.

Proposition 3.3: Consider the DQN algorithm with possibly nonlinear function approximation, and denote $\zeta_n = \nabla_\theta Q^\theta(x_k, u_k)\big|_{\theta=\theta_n}$. Assume that Q^θ is continuously differentiable, and its gradient $\nabla Q^\theta(x, u)$ is globally Lipschitz continuous, with Lipschitz constant independent of (x, u). Suppose that $B = \infty$, the non-negative step-size sequence satisfies $\alpha_n = \alpha_1/n$, with $\alpha_1 > 0$, and suppose that the sequence $\{\theta_n\}$ defined by the DQN algorithm is convergent to some $\theta_\infty \in \mathbb{R}^d$.

Then, this limit is a solution to (12), and moreover the algorithm admits the ODE approximation $\frac{d}{dt}\vartheta_t = \overline{f}(\vartheta_t)$ using the vector field $\overline{f}$ defined in (12). □

The assumption on the step-size is can be replaced by the standard assumptions: $\sum \alpha_n = \infty$, $\sum \alpha_n^2 < \infty$. The proof of Prop. 3.3 can be found in [27]. Its conclusion should raise a warning, since we do not know if (12) has a solution, or if a solution has desirable properties. The conclusions are very different for convex Q-learning.

IV. Examples

BCQL2 for LQR Consider the 3-dimensional LQR model:

$$F = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 0.1 \\ 0.2 \end{bmatrix} \quad S = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad R = 1$$

The Q-function is quadratic in $z = (x^\intercal, u)^\intercal$.

The algorithm was run with batch size 30, $\kappa^\varepsilon = 0.1$, $\alpha_{n+1} = (1+n)^{-0.85}$, and μ selected uniform on a collection of points $\{z^i\}$ for $1 \le i \le 300$ with unit norm in $\mathbb{R}^4$. For (45) we chose

$$\zeta_k(i) = q\Big(\frac{z_k}{\|z_k\|} - z^i\Big), \quad z_k = (x_k(1), x_k(2), x_k(3), u_k)^\intercal,$$

where q is a $N(0, \sigma^2)$ density, with $\sigma^2 = 20/300$. The input used for training was:

$$u_k = -K_\circ x_k + \frac{g_u}{\sqrt{D}} \sum_{i=1}^{D} \sin(\omega_i k)$$

where $K_\circ$ is selected such that eigenvalues of the matrix $(F - GK_\circ)$ lie in the open unit disk. The sequence $\{\omega_i\}$ were chosen as follows:

$$\omega_i = 10 + 100U_i, \quad 1 \le i \le D$$

where $\{U_i\}$ were drawn independently from a uniform distribution in the interval $[0, 1]$. In each experiment, we set $g_u = 4.0$ and $D = 30$. Fig. 1 shows quick convergence to the optimal solution.

CQL for MountainCar Consider the example as formulated in [40, Ch. 10]:

$$\begin{aligned} z_{k+1} &= [\![x_1(t) + x_2(t)]\!]_1 \\ v_{k+1} &= [\![v_k + 10^{-3}u_k - 2.5 \times 10^{-3}\cos(3z_k)]\!]_2 \end{aligned} \quad (48)$$

The brackets are projecting the values of z_{k+1} to the interval $[z^{\min}, z^{\text{goal}}] = [-1.2, 0.5]$, and z_{k+1} to the interval $[-\overline{v}, \overline{v}]$, with $\overline{v} = 7 \times 10^{-2}$. The two dimensional state process $x_k = (z_k, v_k)^\intercal$ evolves in $\mathsf{X} = [z^{\min}, z^{\text{goal}}] \times [-\overline{v}, \overline{v}]$. The control objective is to reach the goal $z^{\text{goal}} = 0.5$ in minimal time.

Fig. 2 shows results from one experiment using the CQL algorithm (37) in which basis functions for J^θ and Q^θ were defined via binning—see [27] for details.

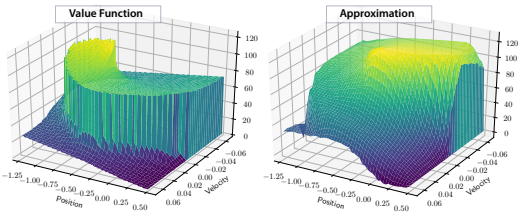

Fig. 2: Value function and its approximation using CQL

V. Conclusions

The LP and QP characterization of dynamic programming equations gives rise to RL algorithms that are provably convergent, and for which we know what problem we are actually solving. Much more work is required to develop these algorithms for particular applications, and to improve efficiency through a combination of algorithm design and techniques from optimization theory. Applications to actor-critic methods is another topic of current research.

References

[1] A. Arapostathis, V. S. Borkar, E. Fernandez-Gaucherand, M. K. Ghosh, and S. I. Marcus. Discrete-time controlled Markov processes with average cost criterion: a survey. *SICON*, 1993.

[2] J. Bas-Serrano, S. Curi, A. Krause, and G. Neu. Logistic Q-learning. In *AISTATS*, 2021.

[3] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *ICML*, pages 449–458, 2017.

[4] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. Mehta, and S. Meyn. Quasi-stochastic approximation and off-policy reinforcement learning. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 5244–5251, Mar 2019.

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

[6] D. P. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[7] V. S. Borkar. Convex analytic methods in Markov decision processes. In *Handbook of Markov decision processes*, volume 40 of *Internat. Ser. Oper. Res. Management Sci.*, pages 347–375. Kluwer, 2002.

[8] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint (2nd ed., to appear)*. Hindustan Book Agency, 2020.

[9] V. S. Borkar, V. Gaitsgory, and I. Shvartsman. LP formulations of discrete time long-run average optimal control problems: The non ergodic case. *SIAM Journal on Control and Optimization*, 57(3):1783–1817, 2019.

[10] V. S. Borkar and S. P. Meyn. The ODE method for convergence of stochastic approximation and reinforcement learning. SICON, 2000.

[11] S. Chen, A. Bernstein, A. Devraj, and S. Meyn. Accelerating optimization and reinforcement learning with quasi-stochastic approximation. To appear, ACC 2021 and arXiv:2009.14431

[12] W. Dabney, M. Rowland, M. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[13] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Res.*, 2003.

[14] D. P. 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.

[15] C. Derman. *Finite State Markovian Decision Processes*, volume 67 of *Mathematics in Science and Engineering*. Academic Press, 1970.

[16] A. M. Devraj, A. Bušić, and S. Meyn. Fundamental design principles for reinforcement learning algorithms. In K. G. Vamvoudakis, Y. Wan, F. L. Lewis, and D. Cansever, editors, *Handbook on Reinforcement Learning and Control*. Springer, 2021.

[17] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *ICML*, pages 1582–1591, 2018.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[19] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI*, volume 32, 2018.

[20] A. Kamoutsi, T. Sutter, P. Mohajerin Esfahani, and J. Lygeros. On infinite linear programming and the moment approach to deterministic infinite horizon discounted optimal control problems. *IEEE Control Systems Letters*, 1(1):134–139, July 2017.

[21] S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[22] D. Lee and N. He. Stochastic primal-dual Q-learning algorithm for discounted MDPs. In *Proc. of the ACC*, 2019.

[23] D. Lee and N. He. A unified switching system perspective and ODE analysis of Q-learning algorithms. arXiv:1912.02270, 2019.

[24] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton. Toward off-policy learning control with function approximation. ICML'10, pages 719–726, 2010.

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

[26] P. G. Mehta and S. P. Meyn. Q-learning and Pontryagin's minimum principle. In *Proc. of the IEEE CDC*, Dec. 2009.

[27] P. G. Mehta and S. P. Meyn. Convex Q-learning, part 1: Deterministic optimal control. *ArXiv e-prints:2008.03559*, 2020.

[28] F. S. Melo, S. P. Meyn, and M. I. Ribeiro. An analysis of reinforcement learning with function approximation. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 664–671, New York, NY, USA, 2008. ACM.

[29] S. Meyn. *Control Systems and Reinforcement Learning*. Cambridge University Press (In preparation), 2021.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing Atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[33] G. Neu and C. Pike-Burke. A unifying view of optimism in episodic reinforcement learning. In *NeurIPS*, 2020.

[34] I. Osband, B. Van Roy, D. Russo, and Z. Wen. Deep exploration via randomized value functions. *arXiv:1703.07608*, 2017.

[35] I. Osband, B. Van Roy, and Z. Wen. Generalization and exploration via randomized value functions. In *ICML*, pages 2377–2386, 2016.

[36] A. Ramaswamy and S. Bhatnagar. Stability of stochastic approximations with 'controlled Markov' noise and temporal difference learning. *IEEE Transactions on Automatic Control*, pages 1–1, 2018.

[37] M. Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[38] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015.

[39] P. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *J. of Math. Anal. and Appl.*, 110:568–582, 1985.

[40] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[41] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[42] L. Vandenberghe and S. Boyd. Applications of semidefinite programming. *Applied Numerical Mathematics*, 29(3):283 – 299, 1999.

[43] R. Vinter. Convex duality and nonlinear optimal control. *SIAM Journal on Control and Optimization*, 31(2):518–21, 03 1993.

[44] Y. Wang and S. Boyd. Performance bounds for linear stochastic control. *Systems Control Lett.*, 58(3):178–182, 2009.

[45] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, pages 1995–2003, 2016.