October 2011 (Revised Feb. 2012) **Report LIDS - 2874**

Lambda-Policy Iteration: A Review and a New Implementation[†]

Dimitri P. Bertsekas [‡]

Abstract

In this paper we discuss λ-policy iteration, a method for exact and approximate dynamic programming. It is intermediate between the classical value iteration (VI) and policy iteration (PI) methods, and it is closely related to optimistic (also known as modified) PI, whereby each policy evaluation is done approximately, using a finite number of VI. We review the theory of the method and associated questions of bias and exploration arising in simulation-based cost function approximation. We then discuss various implementations, which offer advantages over well-established PI methods that use LSPE(λ), LSTD(λ), or TD(λ) for policy evaluation with cost function approximation. One of these implementations is based on a new simulation scheme, called geometric sampling, which uses multiple short trajectories rather than a single infinitely long trajectory.

1. INTRODUCTION

Approximate dynamic programming (DP for short) has attracted substantial research interest, and has a wide range of applications, because of its potential to address large and complex problems that may not be treatable in other ways. The literature on the subject is very extensive, and includes several textbooks, research monographs, and surveys that relate to the computational context of this paper. For a nonexhaustive list, we mention the books by Bertsekas and Tsitsiklis [BeT96], Sutton and Barto [SuB98], Gosavi [Gos03], Cao [Cao07], Chang, Fu, Hu, and Marcus [CFH07], Meyn [Mey07], Powell [Pow07], Borkar [Bor08], Haykin [Hay08], Busoniu, Babuska, De Schutter, and Ernst [BBD10], and the author's text in preparation [Ber11a]; the edited volumes and special issues by White and Sofge [WhS92], Si, Barto, Powell, and Wunsch [SBP04], Lewis, Lendaris, and Liu [LLL08], and the 2007-2009 Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning; and the recent surveys by Borkar [Bor09], Lewis and Vrabie [LeV09], Werbos [Wer09], Szepesvari [Sze10], and Bertsekas [Ber11b].

The purpose of this paper is to critically review and extend a class of methods for exact and approximate DP, which are based on the λ-policy iteration (λ-PI) method, proposed by Bertsekas and Ioffe [BeI96]. This method is intermediate between the classical value iteration (VI) and policy iteration (PI) methods, and

[†] To appear in *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, by F. Lewis and D. Liu (eds.), IEEE Press Computational Intelligence Series.

[‡] The author is with the Dept. of Electr. Engineering and Comp. Science, M.I.T., Cambridge, Mass., 02139. His research was supported by NSF Grant ECCS-0801549, and by the Air Force Grant FA9550-10-1-0412. Thanks are due to Bruno Scherrer for helpful comments, to Huizhen Yu for related collaboration, and to Mengdi Wang for assistance with computational experimentation.

it is closely related to optimistic (also known as modified) PI, whereby each policy evaluation is done approximately, using a finite number of VI. It was originally used as the starting point for the development of approximate simulation-based DP methods of the temporal difference (TD) type, such as LSPE(λ) (see [BeI96], and also [BeT96], Sections 2.3.1 and 8.3). The emphasis in this paper is on implementations of λ-PI, which provide alternatives to approximate PI methods that use other more established methods for policy evaluation.

We will focus on the α-discounted n-state Markovian Decision Problem (MDP), although the main ideas are more broadly applicable. The problem involves states $1, \ldots, n$, controls $u \in U(i)$ at state i, transition probabilities $p_{ij}(u)$, and cost $g(i, u, j)$ for transition from i to j under control u. A (stationary) policy μ is a function from states i to admissible controls $u \in U(i)$, and $J_\mu(i)$ is the cost starting from state i and using policy μ. It is well-known (see e.g., Puterman [Put94] or Bertsekas [Ber07]) that the vector $J_\mu \in \Re^n$, which has components $J_\mu(i)$,† is the unique fixed point of the mapping $T_\mu : \Re^n \mapsto \Re^n$, which maps $J \in \Re^n$ to the vector $T_\mu J \in \Re^n$ that has components

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i, \mu(i), j) + \alpha J(j)\big), \qquad i = 1, \ldots, n. \tag{1.1}$$

Similarly, the optimal costs starting from $i = 1, \ldots, n$, are denoted $J^*(i)$, and the optimal cost vector $J^* \in \Re^n$, which has components $J^*(i)$, is the unique fixed point of the mapping $T : \Re^n \mapsto \Re^n$ defined by

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha J(j)\big), \qquad i = 1, \ldots, n. \tag{1.2}$$

An important property is that T_μ and T are sup-norm contractions. In particular, the iterations $J_{k+1} = T_\mu J_k$ and $J_{k+1} = TJ_k$ converge to J_μ and J^*, respectively, from any starting point J_0 - this is the VI method.

A major alternative to VI is PI. It produces a sequence of policies and associated cost functions through iterations that have two phases: *policy evaluation* (where the cost function of a policy is evaluated), and *policy improvement* (where a new policy is generated). In the exact form of the algorithm, the current policy μ is improved by finding $\bar\mu$ that satisfies $T_{\bar\mu} J_\mu = TJ_\mu$ [i.e., by minimizing in the right-hand side of Eq. (1.2) with J_μ in place of J]. The improved policy $\bar\mu$ is evaluated by solving the linear system of equations $J_{\bar\mu} = T_{\bar\mu} J_{\bar\mu}$, and $(J_{\bar\mu}, \bar\mu)$ becomes the new cost vector-policy pair, which is used to start a new iteration. Thus, the exact form of PI can be succinctly defined as

$$T_{\mu_{k+1}} J_k = TJ_k, \qquad J_{k+1} = T_{\mu_{k+1}} J_{k+1}, \tag{1.3}$$

with the equation on the left describing the policy improvement and the equation on the right describing the evaluation of μ_{k+1}.

In a variant of the method, a policy μ_{k+1} is evaluated by a finite number of applications of $T_{\mu_{k+1}}$ to an approximate evaluation of the preceding policy. This is known as "optimistic" or "modified" PI, and its motivation is that in problems with a large number of states, the linear system $J_{k+1} = T_{\mu_{k+1}} J_{k+1}$ cannot be practically solved directly by matrix inversion, so it is best solved iteratively by VI. The method can be succinctly defined as

$$T_{\mu_{k+1}} J_k = TJ_k, \qquad J_{k+1} = T_{\mu_{k+1}}^{m_k} J_k. \tag{1.4}$$

† In our notation, $\Re^n$ is the n-dimensional Euclidean space, all vectors in $\Re^n$ are viewed as column vectors, and a prime denotes transposition. The identity matrix is denoted by I.

If the number m_k of applications of $T_{\mu_{k+1}}$ is very large, the exact form of PI is essentially obtained, but practice has shown that it is most efficient to use a moderate value of m_k. In this case, the algorithm looks like a hybrid of VI and PI, involving a sequence of alternate applications of T and T_{μ_k}, with μ_k changing over time. Optimistic PI is generally believed to be more computationally efficient that either VI or PI. This is particularly so for problems where n is very large and implementation of exact PI is difficult due to the associated $n \times n$ matrix inversion, and also for problems with a large number of controls, where the overhead due to minimization over all controls $u \in U(i)$ in the mapping T [cf. Eq. (1.2)] is substantial.

We note that the convergence properties of the optimistic PI method (1.4) are quite complicated and have been the subject of continuing research. The convergence $J_k \to J^*$ has been established by Rothblum [Rot79] (see also the more recent work by Canbolat and Rothblum [CaR11], which extends some of the results of [Rot79]). On the other hand, when optimistic PI is implemented asynchronously (as it normally would be when simulation is used), it may oscillate as shown by the convergence counterexamples of Williams and Baird [WiB93]. Recent work of Bertsekas and Yu [BeY10a], [BeY10b], [YuB11] has developed convergent variants of synchronous and asynchronous optimistic PI and Q-learning, based on a new way to perform policy evaluation: by solving approximately an optimal stopping problem rather than a system of linear equations.

The λ-PI method is a form of optimistic PI, given by

$$T_{\mu_{k+1}} J_k = T J_k, \qquad J_{k+1} = T^{(\lambda)}_{\mu_{k+1}} J_k, \tag{1.5}$$

where for any μ and $\lambda \in [0,1)$, $T^{(\lambda)}_{\mu}$ is the linear mapping given by

$$T^{(\lambda)}_{\mu} = (1-\lambda) \sum_{\ell=0}^{\infty} \lambda^{\ell} T_{\mu}^{\ell+1}. \tag{1.6}$$

Note that the mapping $T^{(\lambda)}_{\mu}$ is central in much recent research on approximate DP, simulation-based PI, and TD methods, as will be discussed in the sequel.

To compare the optimistic PI method (1.4) and the λ-PI method (1.5), note that both mappings $T^{m_k}_{\mu_{k+1}}$ and $T^{(\lambda)}_{\mu_{k+1}}$ appearing in Eqs. (1.4) and (1.5), involve multiple applications of the VI mapping $T_{\mu_{k+1}}$: a fixed number m_k in the former case (with $m_k = 1$ corresponding to VI and $m_k \to \infty$ corresponding to PI), and a geometrically weighted number in the latter case (with $\lambda = 0$ corresponding to VI and $\lambda \to 1$ corresponding to PI). Thus optimistic PI and λ-PI are similar: they just control the accuracy of the approximation $J_{k+1} \approx J_{\mu_{k+1}}$ by applying VI in different ways. In a classical DP/non-simulation-based setting, λ-PI is far more complicated relative to optimistic PI, since exact computations using the mapping $T^{(\lambda)}_{\mu}$ are unwieldy. However, this advantage of optimistic PI is dissipated in a simulation context, where computations involving $T^{(\lambda)}_{\mu}$ can be performed conveniently, as extensive analytical and experimental work with TD methods has demonstrated.

Recent research on DP has focused on the use of simulation, in order to deal with model-free situations where the transition probabilities and/or the cost per stage are not known explicitly, and also to deal with the associated high-dimensional linear algebra operations. For problems with very large number of states, the evaluation of various fixed points of mappings, such as T_μ or $T^{(\lambda)}_{\mu}$, is typically done by approximation with a vector Φr from the subspace $S = \{\Phi r \mid r \in \Re^s\}$ that is spanned by the columns of an $n \times s$ matrix Φ. In this paper we will focus on the *projected equation approach*, whereby given a generic mapping $L : \Re^n \mapsto \Re^n$ (such as for example T_μ) we approximate its fixed point by solving the equation

$$\Phi r = \Pi L(\Phi r),$$

where Π denotes projection onto the subspace S. The projection is with respect to a Euclidean norm $\|\cdot\|_\xi$, weighted by a suitable vector ξ of positive weights. An alternative possibility is to solve instead the equation

$$\Phi r = \Pi L^{(\nu)}(\Phi r), \tag{1.7}$$

where, similar to Eq. (1.6),

$$L^{(\nu)} = (1-\nu)\sum_{\ell=0}^{\infty} \nu^\ell L^{\ell+1},$$

and $\nu \in [0,1)$ is a parameter [not necessarily the same as the λ parameter in Eqs. (1.5)-(1.6)]. In our context we will encounter several different types of mappings L, and in all cases L is a contraction with respect to the projection norm $\|\cdot\|_\xi$, with fixed point $\hat{J}$, while $\Pi L^{(\nu)}$ are contractions with respect to $\|\cdot\|_\xi$ for all $\nu \in [0,1)$. It is well-known that the fixed point of $\Pi L^{(\nu)}$, denoted $\Phi r(\nu)$, converges to $\Pi \hat{J}$ as $\nu \to 1$. The norm of the difference $\Phi r(\nu) - \Pi \hat{J}$ is known as the *bias.* Its size/norm depends on ν and is generally smaller as ν gets closer to 1 (see [BeT96], [TsV97], [YuB10] for error bound analyses).

A common example of fixed point approximation in PI is when $L = T_\mu$ for a policy μ, in which case the fixed point of ΠL or $\Pi L^{(\nu)}$ is an approximation to the fixed point of T_μ, i.e., the cost vector J_μ. If the Markov chain corresponding to μ is irreducible and ξ is the corresponding steady-state distribution vector, the mapping $\Pi T_\mu^{(\lambda)}$ is a contraction with respect to $\|\cdot\|_\xi$ for all $\lambda \in [0,1)$, and is unique fixed point, denoted $\Phi r_\mu(\lambda)$, converges to ΠJ_μ as $\lambda \to 1$. Generally, the projected equation $\Phi r = \Pi T_\mu^{(\lambda)}(\Phi r)$ is solved by a simulation process that generates a sequence of states according to a sampling scheme to be discussed later, and then by matrix inversion [this is the Least Squares Temporal Differences [LSTD(λ)] method, proposed by Bradtke and Barto [BrB96]], or by iteration, using the TD(λ) method, proposed by Sutton [Sut87] and analyzed by Tsitsiklis and VanRoy [TsV97] among others, or the Least Squares Policy Evaluation [LSPE(λ)] method, proposed by Bertsekas and Ioffe [BeI96].† These methods are extensively discussed in the literature, and exhibit complex and sometimes pathological behavior, particularly when embedded within PI (see [Ber95], [SzL06], [ThS09] for some notable failures, and [Ber10] for a recent assessment). Moreover matrix inversion and iterative methods, like TD(λ), LSTD(λ), and LSPE(λ), can be used for solving not only the projected equation $\Phi r = \Pi T_\mu^{(\lambda)}(\Phi r)$, but also the more general equation $\Phi r = \Pi L^{(\nu)}(\Phi r)$ of Eq. (1.7), as long as L is a linear mapping that is convenient for the use of simulation [and in the case of TD(λ) and LSPE(λ), $\Pi L^{(\nu)}$ is a contraction; see [BeY09] or [Ber11c]].

In this paper we will review some of the basic issues in approximate PI using the projected equation approach, thereby setting the stage for assessing the relative strengths and weaknesses of the λ-PI methodology. We will then focus on three alternative implementations of λ-PI, which involve simulation and cost

† The paper [BeI96] as well as the book [BeT96] used the name "λ-policy iteration" for both the lookup table and the compact representation versions of the method described here, and tested a compact representation version on the game of tetris, a challenging SSP problem. The name "LSPE" was first used in the subsequent paper by Nedić and Bertsekas [NeB03] to describe a specific iterative implementation of the λ-PI method with cost function approximation for discounted MDP (essentially the discounted version of the implementation used in [BeI96] and [BeT96] for the aforementioned tetris case study). Reference [NeB03] proved convergence of the LSPE(λ) method, as described in Section 3.1, for the case of a diminishing stepsize. Convergence for a stepsize equal to 1 was proved shortly afterwards by Bertsekas, Borkar, and Nedić [BBN04]. The use of two different names for essentially the same method has been a source of some confusion. While in practical implementations these two names refer to algorithms that are closely related, we reserve the name "λ-policy iteration" for the more abstract form (1.5)-(1.6), and we will view LSPE(λ) as an implementation of λ-PI (see Section 4.1).

function approximation. The first is basically the LSPE(λ) method as implemented in [BeI96]. The second is an interesting recent proposal by Thiery and Scherrer [ThS10a], who gave extensive and quite successful computational results, as well as error bounds [ThS10b]. The third implementation is new and may have some advantages over the first two. We will argue that it deals better with the combined issues of bias and exploration. This implementation embodies a new idea for λ-methods: a simulation scheme, called *geometric sampling*, that uses multiple short trajectories with random geometrically distributed length, and exploration-enhanced restart, rather than a single infinitely long trajectory.

The three implementations are described in Section 4, following a discussion of the generic properties of exact λ-PI in Section 2, and the LSTD(λ) and LSPE(λ) methods in Section 3. In our description, these implementations are model-based and use cost function approximation, but there are versions that are model-free and use Q-factor approximation; these can be straightforwardly constructed by the reader.

2. LAMBDA-POLICY ITERATION WITHOUT COST FUNCTION APPROXIMATION

We first recall a central result from [BeI96]. It provides a helpful characterization of the λ-PI method (1.5), which will later become the basis for cost function approximations.

Proposition 2.1: Given $\lambda \in [0,1)$, J_k, and μ_{k+1}, consider the mapping W_k defined by

$$W_k J = (1-\lambda) T_{\mu_{k+1}} J_k + \lambda T_{\mu_{k+1}} J. \tag{2.1}$$

(a) W_k is a sup-norm contraction of modulus $\lambda\alpha$.

(b) The vector $J_{k+1} = T^{(\lambda)}_{\mu_{k+1}} J_k$ generated next by the λ-PI method (1.5) is the unique fixed point of W_k.

Proof: (a) For any two vectors J and $\bar{J}$, using the definition (2.1) of W_k, we have

$$\|W_k J - W_k \bar{J}\| = \big\|\lambda(T_{\mu_{k+1}} J - T_{\mu_{k+1}} \bar{J})\big\| = \lambda \|T_{\mu_{k+1}} J - T_{\mu_{k+1}} \bar{J}\| \le \lambda\alpha \|J - \bar{J}\|,$$

where $\|\cdot\|$ denotes the sup-norm, so W_k is a sup-norm contraction with modulus $\lambda\alpha$.

(b) We have

$$J_{k+1} = T^{(\lambda)}_{\mu_{k+1}} J_k = (1-\lambda) \sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}_{\mu_{k+1}} J_k,$$

so the fixed point property to be shown, $J_{k+1} = W_k J_{k+1}$, is written as

$$(1-\lambda) \sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}_{\mu_{k+1}} J_k = (1-\lambda) T_{\mu_{k+1}} J_k + \lambda T_{\mu_{k+1}} (1-\lambda) \sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}_{\mu_{k+1}} J_k,$$

and evidently holds. **Q.E.D.**

From part (b) of the preceding proposition, we see that $J_{k+1} = W_k J_{k+1}$, or equivalently

$$J_{k+1}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu_{k+1}(i)\big)\Big(g\big(i,\mu_{k+1}(i),j\big) + (1-\lambda)\alpha J_k(j) + \lambda\alpha J_{k+1}(j)\Big), \qquad i = 1,\ldots,n. \tag{2.2}$$

The solution of this fixed point equation can be obtained by viewing it as Bellman's equation for two equivalent MDP.

(a) *As Bellman's equation for an infinite-horizon $\lambda\alpha$-discounted MDP* where μ_{k+1} is the only policy, and the cost per stage is
$$g\big(i,\mu_{k+1}(i),j\big) + (1-\lambda)\alpha J_k(j).$$

(b) *As Bellman's equation for an infinite-horizon stopping problem* where μ_{k+1} is the only policy. In particular, J_{k+1} is the cost vector of policy μ_{k+1} in a stopping problem that is derived from the given α-discounted problem by introducing transitions from each state j to an artificial termination state as follows: at state i we first make a transition to j with probability $p_{ij}\big(\mu_{k+1}(i)\big)$ and transition cost $g\big(i,\mu_{k+1}(i),j\big)$; then we either stay at j and wait for the next transition (this occurs with probability λ), or else we move from j to the termination state with an additional termination cost $\alpha J_k(j)$ (this occurs with probability $1-\lambda$). All transition costs as well as the termination cost are discounted by an additional factor α with each transition.

The convergence and rate of convergence of the λ-PI method (1.5) was given in [BeI96] and also in [BeT96], Prop. 2.8. We will simply quote the results for completeness.

Proposition 2.2: Assume that $\lambda \in [0,1)$, and let $\{J_k, \mu_k\}$ be the sequence generated by the λ-PI method (1.5). Then J_k converges to J^*. Furthermore, for all k greater than some index $\bar{k}$, μ_k is optimal.

Proposition 2.3: Let the assumptions of Prop. 2.2 hold and let $\bar{k}$ be the index such that for all $k \geq \bar{k}$, μ_k is optimal. The sequence $\{J_k\}$ generated by the λ-PI method (1.5) satisfies for all $k > \bar{k}$

$$\|J_{k+1} - J^*\| \leq \frac{\alpha(1-\lambda)}{1-\lambda\alpha}\|J_k - J^*\|, \tag{2.3}$$

where $\|\cdot\|$ denotes the sup-norm.

Note that the convergence rate estimate (2.3) holds only for $k \geq \bar{k}$, essentially after an optimal policy has been identified, as per Prop. 2.2. Nonetheless, this rate estimate is qualitatively correct, and supports the empirical observation that the iterates (J_k, μ_k) generated by λ-PI converge faster as λ increases. Indeed in the limit, as $\lambda \to 1$, λ-PI becomes exact PI, and converges to the optimum in a finite number of iterations.

On the other hand, the computation of $J_{k+1} = T^{(\lambda)}_{\mu_{k+1}} J_k$ [cf. Eq. (1.5)] becomes more time-consuming as λ increases, particularly when simulation is used, because the simulation-based calculation of $T^{(\lambda)}_{\mu_{k+1}} J_k$ involves more simulation noise as λ gets larger.

We finally note that Props. 2.2 and 2.3 apply to synchronous implementations of λ-PI. When implemented asynchronously, λ-PI has similar convergence difficulties to optimistic PI. To see this, note that asynchronous implementations of these two methods essentially coincide when $m_k = 1$ in Eq. (1.5) and $\lambda = 0$ in Eq. (1.4), and the counterexamples of Williams and Baird [WiB93] apply. Thus the development of convergent versions of asynchronous λ-PI is an open research question.

3. APPROXIMATE POLICY EVALUATION USING PROJECTED EQUATIONS

In PI methods with cost function approximation, we evaluate μ by approximating J_μ with a vector Φr_μ from the subspace $S = \{\Phi r \mid r \in \Re^s\}$, spanned by the columns of an $n \times s$ matrix Φ, which may be viewed as basis functions. We generate an "improved" policy $\bar\mu$ using the formula $T_{\bar\mu}(\Phi r_\mu) = T(\Phi r_\mu)$, i.e.,

$$\bar\mu(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha\phi(j)' r_\mu\big), \qquad i = 1,\ldots,n,$$

where $\phi(j)'$ is the row of Φ that corresponds to state j [the method terminates with μ if $T_\mu(\Phi r_\mu) = T(\Phi r_\mu)$]. We then repeat with μ replaced by $\bar\mu$. For the purposes of this paper, we assume that Φ has rank s, and that the Markov chain corresponding to μ is irreducible.

As noted earlier, in the projected equation approach to approximate PI, we approximate J_μ with a vector of the form $\Phi r_\mu(\lambda)$ that solves the fixed point problem

$$\Phi r = \Pi T^{(\lambda)}_\mu(\Phi r). \tag{3.1}$$

Here Π denotes projection onto the subspace S with respect to a weighted Euclidean norm $\|\cdot\|_\xi$, where $\xi = (\xi_1,\ldots,\xi_n)$ is a probability distribution with positive components (i.e., $\|J\|^2_\xi = \sum_{i=1}^n \xi_i x_i^2$, where $\xi_i > 0$ for all i). In nonoptimistic PI methods, the projected equation (3.1) is solved exactly, while in optimistic PI methods it is solved approximately. We note that this approach has a long history in the context of Galerkin methods for the approximate solution of high-dimensional or infinite-dimensional linear equations (partial differential, integral, inverse problems, etc; see e.g., [Kra72], [Fle84]). In fact some of the policy evaluation theory referred to in this paper applies to general projected equations arising in contexts beyond DP (see [BeY09], [Ber09], [Yu10a,b], [Ber11c]). However, Monte Carlo simulation is not part of the Galerkin methodology, as currently practiced in the numerical analysis field. For this reason much of the extensive available knowledge about Galerkin methods does not apply to the approximate DP context, which is primarily simulation-oriented.

We now discuss some of the issues relating to projected equations. While we focus on Eq. (3.1), much of our discussion also applies to the more general projected equations.

Exploration-Contraction Tradeoff

An important choice in the projected equation approach is the distribution ξ that defines the projection norm $\|\cdot\|_\xi$. This distribution is sometimes chosen to be the steady-state probability vector ξ_μ of the Markov chain corresponding to μ, in which case the mapping $\Pi T_\mu^{(\lambda)}$ can be shown to be a contraction with respect to $\|\cdot\|_{\xi_\mu}$ with modulus

$$\alpha_\lambda = \frac{\alpha(1-\lambda)}{1-\lambda\alpha} \tag{3.2}$$

(see [BeT96], Lemma 6.6, or [Ber07], Prop. 6.3.3).

On the other hand the choice of ξ is related to *exploration*, i.e., the need to collect an adequately rich set of samples from a broad and representative set of states. This is a critical issue in simulation-based PI, and results in a well-known tradeoff: to evaluate a policy μ, we may need to generate cost samples using μ, but this may affect the simulation results by underrepresenting states that are unlikely to occur under μ (more weight is placed on states that are visited more frequently under μ). As a result, the cost-to-go estimates of the underrepresented states may be highly inaccurate, causing potentially serious errors in the calculation of the improved control policy.

A well-known approach for exploration is to choose ξ to be a mixture of the form

$$\xi = (1-\beta)\xi_\mu + \beta\tilde{\xi}, \tag{3.3}$$

where $\beta \in (0,1)$ and $\tilde{\xi}$ is another distribution (often referred to as the *off-policy* distribution), which is added to enhance exploration (see the discussion of Section 1). Unfortunately, with such a choice the contraction property of $\Pi T_\mu^{(\lambda)}$ comes into doubt: it depends on the size of the parameters λ and β [it can be shown that $\Pi T_\mu^{(\lambda)}$ is a contraction for any $\beta \in [0,1)$ provided λ is close enough to 1, and it is a contraction for any $\lambda \in [0,1)$ provided β is close enough to 0]. This is important because for convergence of iterative methods such as TD(λ) and some forms of LSPE(λ), it is critical that $\Pi T_\mu^{(\lambda)}$ be a contraction. Thus there is a tradeoff between exploration enhancement using the mixture distribution (3.3) and ability to use a broader range of methods for solution of the projected equation.

Bias

While the Bellman equation $J = T_\mu^{(\lambda)} J$ has the same fixed point J_μ for all $\lambda \in [0,1)$, the fixed point $\Phi r_\mu(\lambda)$ of the projected version (3.1) depends on λ. The difference of $\Phi r_\mu(\lambda)$ and the closest point of S to J_μ, $\Phi r_\mu(\lambda) - \Pi J_\mu$, is generally nonzero. Its norm, the bias, tends to decrease to 0 as $\lambda \uparrow 1$ and tends to increase as $\lambda \downarrow 0$. It is known that the bias can be very large and may seriously degrade the practical value of the approximate policy evaluation for small values of λ; see [Ber95] for some examples.

The following is a well-known error bound for the case $\xi = \xi_\mu$:

$$\|J_\mu - \Phi r_\mu(\lambda)\|_{\xi_\mu} \le \frac{1}{\sqrt{1-\alpha_\lambda^2}}\,\|J_\mu - \Pi J_\mu\|_{\xi_\mu}, \tag{3.4}$$

where α_λ is given by Eq. (3.2), and $\|\cdot\|_{\xi_\mu}$ is the weighted Euclidean norm corresponding to $\xi = \xi_\mu$, the steady-state probability vector of the Markov chain corresponding to μ. Thus the error bound becomes worse as λ decreases (and α_λ increases), suggesting a larger size of bias. While the bound is rather conservative, the paper by Yu and Bertsekas [YuB10] (see also Scherrer [Sch10]) derives sharper error bounds, which also apply to cases where $\xi \neq \xi_\mu$ and $\Pi T_\mu^{(\lambda)}$ is not a contraction. These error bounds and the bound (3.4) are consistent in suggesting that the bias increases as λ decreases, and they are also largely consistent with the results of computational experimentation.

Bias-Variance Tradeoff

In simulation-based methods for solving the projected equation (3.1), one must deal with the effects of simulation error. Generally as λ increases, the methods become more vulnerable to simulation noise, and hence require more sampling for good performance. Indeed, the noise in a simulation sample of an ℓ-stages cost vector $T_\mu^\ell J$ tends to be larger as ℓ increases, and from the formula

$$T_\mu^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^\ell T_\mu^{\ell+1}$$

it can be seen that simulation samples of $T_\mu^{(\lambda)}(\Phi r_k)$ tend to contain more noise as λ increases. This is consistent with practical experience, and gives rise to the so called bias-variance tradeoff: a large value of λ to reduce bias results in slower and less reliable computation because of higher simulation noise (and consequently, a larger number of samples to achieve the same accuracy of various simulation-based estimates). Generally, there is no rule of thumb for selecting λ, which is usually chosen with some trial and error.

In summary, the preceding discussion suggests that if simulation noise is not an issue (i.e., one can afford many simulation samples) one should choose large values of λ, since then the bias is reduced and one may afford greater exploration without losing the contraction property of $\Pi T_\mu^{(\lambda)}$. In the contrary case, however, the degradation of the estimate of J_μ due to simulation noise may offset whatever bias/contraction benefits a large value of λ may bring.

3.1 TD Methods

Most of the simulation-based methods for solving the projected equation use explicitly or implicitly the notion of temporal difference (TD), which originated in reinforcement learning with the works of Samuel [Sam59], [Sam67] on a checkers-playing program. The first TD method is TD(λ), which can be viewed as an iterative stochastic approximation-type algorithm. The LSTD(λ) method is based on batch simulation: it first generates a batch of state and cost samples, it approximates the projected equation $\Phi r = \Pi T_\mu^{(\lambda)}(\Phi r)$ using these samples, and then solves the equation directly by matrix inversion. Another TD method is LSPE(λ), which while being more iterative, shares much of the simulation philosophy of LSTD(λ).

To describe more specifically the LSTD(λ) and LSPE(λ) methods, we first note that the orthogonality condition that characterizes the projection in the projected equation $\Phi r = \Pi T_\mu^{(\lambda)}(\Phi r)$ is

$$\Phi'\Xi\big(\Phi r - T_\mu^{(\lambda)}(\Phi r)\big) = 0, \tag{3.5}$$

where Ξ is the diagonal matrix with the vector ξ along the diagonal (see e.g., [Ber07]). Thus the projected equation (3.1) is equivalent to the lower-dimensional equation (3.5), which can in turn be written in matrix form as

$$C^{(\lambda)} r = d^{(\lambda)}, \tag{3.6}$$

with

$$C^{(\lambda)} = \Phi'\Xi\big(I - P_\mu^{(\lambda)}\big)\Phi, \qquad d^{(\lambda)} = \Phi'\Xi g_\mu^{(\lambda)}, \tag{3.7}$$

and

$$P_\mu^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^\ell \alpha^{\ell+1} P_\mu^{\ell+1}, \qquad g_\mu^{(\lambda)} = \sum_{\ell=0}^{\infty} \lambda^\ell \alpha^\ell P_\mu^\ell g_\mu, \tag{3.8}$$

where P_μ and g_μ are the transition probability matrix and expected single-stage cost vector corresponding to μ. The LSTD(λ) and LSPE(λ) methods use simulation-based approximations of $C^{(\lambda)}$ and $d^{(\lambda)}$. This is done by simulating a state sequence $(i_0, \ldots, i_t)$ and corresponding transition cost sequence, using the current policy μ (perhaps with exploration enhancement, as discussed earlier). Then after each simulated state i_ℓ, $\ell = 0, \ldots, t$, is generated, estimates $C_\ell^{(\lambda)}$ and $d_\ell^{(\lambda)}$ are obtained using the simulation samples up to time ℓ, using formulas that we will not give here, as they are not important for our purposes. Such formulas, in various alternative forms, can be found in several sources, including the textbooks cited earlier. The papers [NeB03], [BeY09], [Yu10a], [Yu10b] discuss the conditions for the convergence $\lim_{\ell\to\infty} C_\ell^{(\lambda)} = C^{(\lambda)}$, $\lim_{\ell\to\infty} d_\ell^{(\lambda)} = d^{(\lambda)}$ to hold with probability 1.

The LSTD(λ) method is based on simple matrix inversion: after the last state i_t of the simulation trajectory is generated, it computes the solution

$$\hat{r} = \big(C_t^{(\lambda)}\big)^{-1} d_t^{(\lambda)} \tag{3.9}$$

of the corresponding simulation-based approximation to Eq. (3.6),

$$C_t^{(\lambda)} r = d_t^{(\lambda)}, \tag{3.10}$$

and approximates the cost vector J_μ by $\Phi\hat{r}$. An important point is that $\hat{r}$ can be obtained regardless of whether $\Pi T_\mu^{(\lambda)}$ is a contraction. It is only required that $C_t^{(\lambda)}$ is invertible, a much less restrictive condition.

One version of the LSPE(λ) method consists of iterative solution of the system (3.10). It approximates the cost vector J_μ by Φr_{t+1}, where r_{t+1} is obtained at the last step of the iteration

$$r_{\ell+1} = r_\ell - \gamma G_\ell\big(C_\ell^{(\lambda)} r_\ell - d_\ell^{(\lambda)}\big), \qquad \ell = 0, \ldots, t, \tag{3.11}$$

where r_0 is some initial vector, likely the vector obtained from the preceding policy evaluation, γ is a positive stepsize, G_ℓ is the matrix

$$G_\ell = \left(\frac{1}{\ell}\sum_{m=0}^{\ell-1} \phi(i_m)\phi(i_m)'\right)^{-1}, \qquad \ell = 0, \ldots, t, \tag{3.12}$$

and as earlier, $\phi(i)'$ denotes the ith row of the matrix Φ. In the original proposal of [BeI96] the stepsize is $\gamma = 1$; convergence of Φr_t to the fixed point of $\Pi T_\mu^{(\lambda)}$ for this stepsize was shown in [BBN04]. The matrix G_ℓ is a simulation-based approximation of $(\Phi'\Xi\Phi)^{-1}$ (alternative choices of G_ℓ have been discussed recently in [Ber11b], [Ber11c]). There is also an equivalent implementation of this iteration, which is based on solution of a least squares problem (see Section 4.1).

The choice (3.12) for G_ℓ and the use of $\gamma = 1$ are based on a view of the method as an approximation to the projected value iteration method

$$\Phi r_{\ell+1} = \Pi T_\mu^{(\lambda)}(\Phi r_\ell),$$

which after some calculation can be written as

$$\Phi r_{\ell+1} = \Phi\big(r_\ell - (\Phi'\Xi\Phi)^{-1}\big(C^{(\lambda)} r_\ell - d^{(\lambda)}\big)\big),$$

or equivalently, since Φ has full rank, as

$$r_{\ell+1} = r_\ell - (\Phi'\Xi\Phi)^{-1}\big(C^{(\lambda)} r_\ell - d^{(\lambda)}\big);$$

cf. Eq. (3.11)-(3.12) with $\gamma = 1$.

Note that the matrix inversion in Eq. (3.12) is not so onerous, because it can be formed incrementally, with a rank-one correction as each sample becomes available. On the other hand, contrary to LSTD(λ) [and similar to TD(λ)], the LSPE(λ) method (3.11)-(3.12) requires that $\Pi T_\mu^{(\lambda)}$ be a contraction for convergence. Indeed if the simulation is performed using the steady-state distribution ξ_μ, it can be shown that $\Pi T_\mu^{(\lambda)}$ is a contraction, but if the simulation is performed using a mixture/off-policy distribution (3.3) for the purpose of exploration-enhancement, the contraction property may be lost and repeated iterations of the form (3.11) may diverge.

We finally note that in iteration (3.11) the underlying assumption is that we update r as simulation samples are collected and used to form ever improving approximations to C and d. An alternative is to use batch simulation, like in LSTD: first simulate to obtain $C_t^{(\lambda)}$, $d_t^{(\lambda)}$, and G_t, and then solve the system $C_t^{(\lambda)} r = d_t^{(\lambda)}$ iteratively rather than through the direct matrix inversion (3.9), by using any number of iterations of the type (3.11). In fact, we may use *only one* iteration, in which case the method takes the form

$$r_1 = r_0 - \gamma G_t\big(C_t^{(\lambda)} r_0 - d_t^{(\lambda)}\big). \tag{3.13}$$

A single (or very few) iterations may be sufficient if λ is close to 1, since then the contraction modulus of $\Pi T_\mu^{(\lambda)}$ is close to 0 (see e.g., [BeT96], Lemma 6.6, or [Ber07], Prop. 6.3.3), so a single iteration with $\Pi T_\mu^{(\lambda)}$ is very effective, yielding a vector that is close to its fixed point. We will return to this variant of the method later.

3.2 Comparison of LSTD(λ) and LSPE(λ)

There has been speculation about the relative merits of LSTD(λ) and LSPE(λ). Generally speaking, it is difficult to reach definitive conclusions, as there are several complex factors to consider, such as the length of the simulation sequence $(i_0, \ldots, i_t)$, and the potential near-singularity of $C^{(\lambda)}$, which affects the error in the matrix inversion in the LSTD(λ) formula (3.9). As an illustration, consider a few different situations:

(a) Assume, as an idealization, that an infinite number of samples is collected. Then both methods yield in the limit the same result, the fixed point of the projected equation $J = \Pi T_\mu^{(\lambda)} J$. However, in contrast to LSTD(λ), in order to guarantee convergence, LSPE(λ) requires that $\Pi T_\mu^{(\lambda)}$ is a contraction, which interferes with the freedom to do exploration, as discussed earlier.

(b) Assume that $C^{(\lambda)}$ is invertible, but is nearly singular. Then the matrix inversion in the LSTD(λ) formula (3.9) may require a very large number of samples to yield a reasonably accurate solution of $C^{(\lambda)} r = d^{(\lambda)}$.† To correct the sensitivity of LSTD(λ) to simulation noise, it may be necessary to turn

† It is well-known from fundamental error analyses of linear equation solvers that small errors in a nearly singular matrix $C^{(\lambda)}$ will cause large errors in the solution of $C^{(\lambda)} r = d^{(\lambda)}$. Near-singularity of $C^{(\lambda)}$ may be due either to the columns of Φ being nearly linearly dependent or to the matrix $\Xi(I - \alpha P^{(\lambda)})$ being nearly singular [cf. Eq. (3.7)]. Near-linear dependence of the columns of Φ will not affect the error in the solution of the high-dimensional projected equation, which can be written as $\Phi C^{(\lambda)} r = \Phi d^{(\lambda)}$. The reason is that this error depends only on the subspace S and not its representation in terms of the matrix Φ. In particular, if we replace Φ with a matrix ΦB where B is an $s \times s$ invertible scaling matrix, the subspace S will be unaffected and the error in the solution of the projected equation will also be unaffected. On the other hand, near singularity of the matrix $I - \alpha P^{(\lambda)}$ may affect significantly the error. Note that $I - \alpha P^{(\lambda)}$ is nearly singular in the case where α is very close to 1, or in the corresponding undiscounted

it into an iterative method through some form of regularization, which then brings it close to a form of LSPE(λ) (see [Ber09], [WPB09], [Ber11a], [Ber11b], [Ber11c] for such regularization methods and their connection to LSPE). Of course, the situation becomes even more complex if $C^{(\lambda)}$ is singular, perhaps due to inadvertent rank deficiency of Φ (see [WaB11a], [WaB11b] for a discussion of this possibility).

(c) When LSTD(λ) and LSPE(λ) are embedded within a PI framework, the number of samples collected using any one policy is often relatively small. Then the behavior of the two methods becomes very complicated, and it is hard to reach any kind of reliable conclusion [Ber10]. Computational studies indicate that LSPE(λ) being an iterative method, is less sensitive to the matrix inversion errors that afflict LSTD(λ) in the presence of high simulation noise.

The preceding discussion is also relevant to the implementations of λ-PI to be discussed in the next section, since these implementations bear strong relations to both LSTD(λ) and LSPE(λ).

4. LAMBDA-POLICY ITERATION WITH COST FUNCTION APPROXIMATION

We saw in Section 2 that the policy evaluation portion of λ-PI,

$$J_{k+1} = T^{(\lambda)}_{\mu_{k+1}} J_k, \tag{4.1}$$

[cf. Eq. (1.5)] can be implemented in two ways:

(1) By computing $T^{(\lambda)}_{\mu_{k+1}} J_k$.

(2) By finding the fixed point of the mapping W_k [cf. Eq. (2.1)] through solution of the equation

$$J = W_k J, \tag{4.2}$$

which can be viewed as Bellman's equation associated with the current policy for the two equivalent DP problems discussed in Section 2 [cf. Eq. (2.2)]: a $\lambda\alpha$-discounted problem and a stopping problem.

Let us now consider approximation of λ-PI on the subspace $S = \{\Phi r \mid r \in \Re^s\}$. A natural possibility is to introduce projection in the preceding approaches. In particular, we may approximate the λ-PI iterate J_{k+1} of Eq. (4.1) by Φr_{k+1} in three ways:

(a) By using a single projected value iteration for the original α-discounted problem,

$$\Phi r_{k+1} = \Pi T^{(\lambda)}_{\mu_{k+1}} (\Phi r_k). \tag{4.3}$$

This is the original proposal of [BeI96]. It is the variant of the LSPE(λ) method (3.11)-(3.12), which involves just the last iteration.

(b) By solving a projected version of Eq. (4.2), viewing it as Bellman's equation for the $\lambda\alpha$-discounted problem of Section 2, and setting r_{k+1} equal to its solution. This is the proposal of [ThS10a], and implements by simulation the solution of this projected equation, essentially by applying LSTD(0) to Bellman's equation for the $\lambda\alpha$-discounted problem formulated in Section 2.

case where $\alpha = 1$ and P is substochastic with some eigenvalues very close to 1. Large variations in the size of the diagonal components of Ξ may also affect significantly the error, although this dependence is complicated by the fact that Ξ appears not only in the formula $C^{(\lambda)} = \Phi'\Xi(I - \alpha P^{(\lambda)})\Phi$ but also in the formula $d^{(\lambda)} = \Phi'\Xi g^{(\lambda)}$.

(c) By solving a projected version of Eq. (4.2), viewing it as Bellman's equation for the stopping problem formulated in Section 2, and setting r_{k+1} equal to its solution.

In the following three subsections, we will describe three alternative implementations of λ-PI corresponding to the possibilities (a)-(c) above. Of course when linear cost function approximation of the form Φr_k is used to represent J_k, the λ-PI method need not converge, and the cost vectors J_{μ_k} of the generated policies typically oscillate within some suboptimality threshold from J^*. We do not address this issue, but we note that related error bounds, which also apply to other forms of optimistic PI are given by Bertsekas and Yu [BeY10a], Thiery and Scherrer [ThS10b], and Scherrer [Sch11].

4.1 The LSPE(λ) Implementation

A variant of the LSPE(λ) method (3.11)-(3.12) is to form batches of simulation samples and perform iteration (3.11) at the end of each batch. In an extreme case, we treat the entire simulation trajectory $(i_0, \ldots, i_t)$ as a single simulation batch, and we perform a *single* iteration (3.11), for $\ell = t$, yielding the method

$$r_{k+1} = r_k - G_t\big(C_t^{(\lambda)} r_k - d_t^{(\lambda)}\big), \tag{4.4}$$

where Φr_k is the approximate evaluation of the cost vector of the preceding policy μ_k [cf. Eq. (3.13)]. As $t \to \infty$ and the simulation becomes exact in the limit, i.e.,

$$\lim_{t\to\infty} C_t^{(\lambda)} = C^{(\lambda)}, \qquad \lim_{t\to\infty} d_t^{(\lambda)} = d^{(\lambda)},$$

and if G_t is given by the formula (3.12), it can be verified that

$$\Phi r_{k+1} \to \Pi T_{\mu_{k+1}}^{(\lambda)}(\Phi r_k). \tag{4.5}$$

Thus the method (4.4) with G_t given by Eq. (3.12) can be viewed as a simulation-based implementation of Eq. (4.3), the projected version of λ-PI, which becomes exact in the limit as $t \to \infty$. In practice of course t is finite, and one may consider variants of the method, whereby multiple iterations of the form (4.4) are performed, with each iteration using additional simulation samples.

We note a mathematically equivalent description of this method, which is given in terms of a least-squares optimization (see [Ber07], Section 6.3.3 for a more detailed textbook account): we set

$$r_{k+1} = \arg\min_{r\in\Re^s} \sum_{\ell=0}^{t} \left(\phi(i_\ell)'r - \phi(i_\ell)'r_k - \sum_{m=\ell}^{t} (\lambda\alpha)^{m-\ell} q(i_m, i_{m+1}) \right)^2, \tag{4.6}$$

where $q(i_m, i_{m+1})$ is the temporal difference

$$q(i_m, i_{m+1}) = g\big(i_m, \mu_{k+1}(i_m), i_{m+1}\big) + \alpha\phi(i_{m+1})'r_k - \phi(i_m)'r_k, \qquad m = 0, \ldots, t. \tag{4.7}$$

In fact this is how the method was originally described in [BeI96] and [BeT96].

A positive aspect of this method is that it approximates directly $\Pi T_{\mu_{k+1}}^{(\lambda)}(\Phi r_k)$, so it is not subject to bias in the evaluation of the fixed point of W_k; cf. Eq. (4.5). However, in the form given here, the method does not address the issue of exploration. Despite this fact, this implementation [in the form (4.6)] has been successful in several challenging computational studies, including the one involving the game of tetris in the original paper [BeI96] and some followup works, and a recent one by Foderaro et. al. [FRF11] involving the game of pac-man, a benchmark problem of pursuit-evasion.

4.2 λ-PI(0) - An Implementation Based on a Discounted MDP

This implementation, suggested and tested by Thiery and Scherrer [ThS10a], [ThS10b], is based on the fixed point property of J_{k+1} [cf. Prop. 2.1(b)]. It produces an approximation Φr_{k+1} to J_{k+1} within the subspace S, by solving the projected equation

$$\Phi r = \Pi W_k(\Phi r),$$

with W_k given by

$$W_k J = (1-\lambda)T_{\mu_{k+1}}(\Phi r_k) + \lambda T_{\mu_{k+1}} J, \qquad J \in \Re^n. \tag{4.8}$$

We may find the solution r_{k+1} of this equation by using an LSTD(0)-like simulation approach. In particular, r_{k+1} satisfies the orthogonality condition

$$Cr = d(k),$$

where

$$C = \Phi'\Xi(I - \lambda\alpha P_{\mu_{k+1}})\Phi, \qquad d(k) = \Phi'\Xi\big(g_{\mu_{k+1}} + (1-\lambda)\alpha P_{\mu_{k+1}}\Phi r_k\big),$$

so that

$$r_{k+1} = C^{-1}d(k). \tag{4.9}$$

We refer to this method as λ-PI(0) to distinguish it notationally from the method of the next subsection (the name LSλPI was introduced for this method in [ThS10a]).

In a simulation-based implementation, the matrix C and the vector $d(k)$ are approximated by estimates C_t and $d_t(k)$. Thus this method does not require that $\Pi T^{(\lambda)}_{\mu_{k+1}}$ is a contraction, and like LSTD, it can deal well with the issue of exploration. The simulation samples need not depend on the policy μ_{k+1} being evaluated, so they can be generated only once within a PI process. On the other hand the objective of the implementation is to approximate the next iterate of λ-PI, i.e., $T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)$, and it is not clear that it is doing this well. To see this, suppose that the iteration (4.9), or equivalently $\Phi r_{k+1} = \Pi W_k(\Phi r_k)$, is repeated an infinite number of times so it converges to a limit $\bar r$, which must satisfy $\Phi\bar r = \Pi W_k(\Phi\bar r)$. Then using Eq. (4.8), we have

$$\Phi\bar r = (1-\lambda)\Pi T_{\mu_{k+1}}(\Phi\bar r) + \lambda\Pi T_{\mu_{k+1}}(\Phi\bar r),$$

which shows that $\Phi\bar r = \Pi T_{\mu_{k+1}}(\Phi\bar r)$. Thus λ-PI(0) aims at $\bar r$, which is the limit of TD(0) independent of the value of λ. Indeed as $\lambda \to 1$, ΠW_k tends to $\Pi T_{\mu_{k+1}}$ [cf. Eq. (4.8)], so its fixed point Φr_{k+1} tends to the fixed point of $\Pi T_{\mu_{k+1}}$, i.e., the limit of TD(0). It follows that while this implementation deals well with the issue of exploration, it may be subject to significant bias-related error.

4.3 λ-PI(1) - An Implementation Based on a Stopping Problem

The third implementation is based on the property mentioned in Section 2: the fixed point equation $J = W_kJ$ [or equivalently, Eq. (2.2)] is Bellman's equation for the policy μ_{k+1} in the context of a stopping problem. Here there is an artificial termination state 0, and for all states j, there is probability $1-\lambda$ that a transition to j will be followed by an immediate transition to state 0, with cost $\alpha J_k(j)$, cf. Eq. (2.2). Note that if λ is not too close to 1, the trajectories of this problem tend to be short, and in fact if $\lambda = 0$ all trajectories consist of a single transition.

To compute an approximation Φr_{k+1} to the fixed point of W_k by using the stopping problem, we may use any policy evaluation algorithm with cost function approximation over the subspace $S = \{\Phi r \mid r \in \Re^s\}$.

An interesting choice is to use the LSPE(1) method, which consists of a least squares fit of Φr to the simulated costs of the trajectories of the stopping problem whose Bellman equation mapping is W_k. The use of LSPE(1) not only involves minimum bias relative to all LSPE(ν) methods with $\nu \in [0,1]$, but also leads to a simple least squares implementation.

To this end, we introduce a simulation procedure, called *geometric sampling*, which departs from the single infinitely long simulation trajectory format of the implementation of Section 4.1, and has the following characteristics:

(a) It uses multiple relatively short simulation trajectories.

(b) The initial state of each trajectory is chosen essentially as desired, thereby allowing flexibility to generate a richer mixture of state visits.

(c) The length of each trajectory is random and is determined by a λ-dependent geometric distribution [a probability $(1-\lambda)\lambda^\ell$ that the number of transitions is $\ell+1$].

In particular, given the current representation Φr_k of J_k and the current policy μ_{k+1}, we update the parameter vector from r_k to r_{k+1} after generating t simulated trajectories. The states of a trajectory are generated according to the transition probabilities $p_{ij}\big(\mu_{k+1}(i)\big)$, the transition cost is discounted by an additional factor α with each transition, and following each transition to a state j, the trajectory is terminated with probability $1-\lambda$ and with an extra cost $\alpha\phi(i)'r_k$. Once a trajectory is terminated, an initial state for the next trajectory is chosen according to a fixed probability distribution $\zeta_0 = \big(\zeta_0(1),\ldots,\zeta_0(n)\big)$, and the process is repeated. Note that the sequence of restart states need not depend on the policy being evaluated, so that it can be simulated only once within a PI process. Of course, the simulated trajectories have to be recalculated for each new policy. The details are as follows.

Let the mth trajectory, $m=1,\ldots,t$, have the form $(i_{0,m}, i_{1,m},\ldots,i_{N_m,m})$, where $i_{0,m}$ is the initial state, and $i_{N_m,m}$ is the state at which the trajectory is completed (the last state prior to termination). For each state $i_{\ell,m}$, $\ell=0,\ldots,N_m-1$, of the mth trajectory, the simulated cost is

$$c_{\ell,m}(r_k) = \alpha^{N_m-\ell}\phi(i_{N_m,m})'r_k + \sum_{q=\ell}^{N_m-1} \alpha^{q-\ell} g(i_{q,m}, u_{q,m}, i_{q+1,m}), \tag{4.10}$$

where

$$u_{q,m} = \mu_{k+1}(i_{q,m}), \qquad m=1,\ldots,t,\ q=0,\ldots,N_m-1.$$

Once the costs $c_{\ell,m}(r_k)$ are computed for all states $i_{\ell,m}$ of the mth trajectory and all trajectories $m=1,\ldots,t$, the vector r_{k+1} is obtained by a least squares fit of these costs:

$$r_{k+1} = \arg\min_{r\in\Re^s} \sum_{m=1}^{t}\sum_{\ell=0}^{N_m-1} \big(\phi(i_{\ell,m})'r - c_{\ell,m}(r_k)\big)^2, \tag{4.11}$$

cf. Eqs. (4.6)-(4.7). Equivalently, we can write the solution of the least squares problem explicitly as

$$r_{k+1} = \left(\sum_{m=1}^{t}\sum_{\ell=0}^{N_m-1} \phi(i_{\ell,m})\phi(i_{\ell,m})'\right)^{-1} \sum_{m=1}^{t}\sum_{\ell=0}^{N_m-1} \phi(i_{\ell,m})c_{\ell,m}(r_k). \tag{4.12}$$

We refer to the resulting implementation as λ-PI(1).

Note the extreme special case when $\lambda=0$. Then all the simulated trajectories consist of a single transition, and there is a restart at every transition. This means that the simulation samples are from states that are generated independently according to the restart distribution ζ_0.

Convergence of the Simulation Process

We will now show that in the limit, as $t \to \infty$, the vector r_{k+1} of Eq. (4.12) satisfies

$$\Phi r_{k+1} = \hat{\Pi} T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k), \tag{4.13}$$

where $\hat{\Pi}$ denotes projection with respect to the weighted sup-norm $\|\cdot\|_\zeta$ with weight vector $\zeta = \big(\zeta(1), \ldots, \zeta(n)\big)$, where

$$\zeta(i) = \frac{\hat{\zeta}(i)}{\sum_{j=1}^n \hat{\zeta}(j)}, \qquad i = 1, \ldots, n, \tag{4.14}$$

and

$$\hat{\zeta}(i) = \sum_{\ell=0}^{\infty} \zeta_\ell(i),$$

with $\zeta_\ell(i)$ being the probability of the state being i after ℓ transitions of a randomly chosen simulation trajectory. This is the underlying norm in TD methods such as LSTD, LSPE, and TD, as applied to SSP problems (see [BeT96], Section 6.3.4). Note that $\zeta(i)$ is the long-term occupancy probability of state i during the simulation process. We assume that the restart distribution ζ_0 is chosen so that $\zeta(i) > 0$ for all $i = 1, \ldots, n$, implying that $\|\cdot\|_\zeta$ is a legitimate norm [this is guaranteed if we require that $\zeta_0(i) > 0$ for all i].

Indeed, let us view $T^{\ell+1}_{\mu_{k+1}} J$ as the vector of total discounted costs over a horizon of $(\ell+1)$ stages with the terminal cost function being J, and write

$$T^{\ell+1}_{\mu_{k+1}} J = \alpha^{\ell+1} P^{\ell+1}_{\mu_{k+1}} J + \sum_{q=0}^{\ell} \alpha^q P^q_{\mu_{k+1}} g_{\mu_{k+1}},$$

where $P_{\mu_{k+1}}$ and $g_{\mu_{k+1}}$ are the transition probability matrix and cost vector, respectively, under μ_{k+1}. As a result the vector $T^{(\lambda)}_{\mu_{k+1}} J = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}_{\mu_{k+1}} J$ can be expressed as

$$\big(T^{(\lambda)}_{\mu_{k+1}} J\big)(i) = \sum_{\ell=0}^{\infty} (1-\lambda)\lambda^\ell E\left\{ \alpha^{\ell+1} J(i_{\ell+1}) + \sum_{q=0}^{\ell} \alpha^q g\big(i_q, \mu_{k+1}(i_q), i_{q+1}\big) \;\Big|\; i_0 = i \right\}, \qquad i = 1, \ldots, n.$$

Thus $\big(T^{(\lambda)}_{\mu_{k+1}} J\big)(i)$ may be viewed as the expected value of the $(\ell+1)$-stages cost of policy μ_{k+1} starting at state i, with *the number of stages being random and geometrically distributed with parameter* λ [probability of $\kappa+1$ transitions is $(1-\lambda)\lambda^\kappa$, $\kappa = 0, 1, \ldots$]. It follows that the cost samples $c_{\ell,m}(r_k)$ of Eq. (4.10), produced by the simulation process described earlier, can be used to estimate $\big(T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)\big)(i)$ for all i by Monte Carlo averaging. The estimation formula is

$$D_t(i) = \frac{1}{\sum_{m=1}^t \sum_{\ell=0}^{N_m-1} \delta(i_{\ell,m} = i)} \cdot \sum_{m=1}^{t} \sum_{\ell=0}^{N_m-1} \delta(i_{\ell,m} = i) c_{\ell,m}(r_k), \tag{4.15}$$

where $\delta(i_{\ell,m} = i) = 1$ if $i_{\ell,m} = i$ and $\delta(i_{\ell,m} = i) = 0$ otherwise, and we have

$$\big(T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)\big)(i) = \lim_{t\to\infty} D_t(i), \qquad i = 1, \ldots, n,$$

(see also the discussion on the consistency of Monte Carlo simulation for policy evaluation in [BeT96], Section 5.2).

Let us now compare the λ-PI iteration (4.13) with the simulation-based implementation (4.12). Using the definition of projection, Eq. (4.13) can be written as

$$r_{k+1} = \arg\min_{r\in\Re^s} \sum_{i=1}^n \zeta(i)\Big(\phi(i)'r - \big(T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)\big)(i)\Big)^2,$$

or equivalently

$$r_{k+1} = \left(\sum_{i=1}^n \zeta(i)\phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^n \zeta(i)\phi(i)\big(T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)\big)(i). \tag{4.16}$$

Let $\tilde\zeta(i)$ be the empirical relative frequency of state i during the simulation, given by

$$\tilde\zeta(i) = \frac{1}{N_1+\cdots+N_t}\sum_{m=1}^t \sum_{\ell=0}^{N_m-1} \delta(i_{\ell,m}=i). \tag{4.17}$$

Then the simulation-based estimate (4.12) can be written as

$$\begin{aligned}
r_{k+1} &= \left(\sum_{m=1}^t \sum_{\ell=0}^{N_m-1} \phi(i_{\ell,m})\phi(i_{\ell,m})'\right)^{-1} \sum_{m=1}^t \sum_{\ell=0}^{N_m-1} \phi(i_{\ell,m})c_{\ell,m}(r_k)\\
&= \left(\sum_{i=1}^n\sum_{m=1}^t \sum_{\ell=0}^{N_m-1} \delta(i_{\ell,m}=i)\phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^n\sum_{m=1}^t \sum_{\ell=0}^{N_m-1} \delta(i_{\ell,m}=i)\phi(i)c_{\ell,m}(r_k)\\
&= \left(\sum_{i=1}^n \tilde\zeta(i)\phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^n \frac{1}{N_1+\cdots+N_t}\cdot\phi(i)\cdot\sum_{m=1}^t \sum_{\ell=0}^{N_m-1} \delta(i_{\ell,m}=i)c_{\ell,m}(r_k)\\
&= \left(\sum_{i=1}^n \tilde\zeta(i)\phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^n \frac{\sum_{m=1}^t\sum_{\ell=0}^{N_m-1}\delta(i_{\ell,m}=i)}{N_1+\cdots+N_t}\cdot\phi(i)\cdot\\
&\qquad\qquad\cdot\frac{1}{\sum_{m=1}^t\sum_{\ell=0}^{N_m-1}\delta(i_{\ell,m}=i)}\cdot\sum_{m=1}^t \sum_{\ell=0}^{N_m-1} \delta(i_{\ell,m}=i)c_{\ell,m}(r_k)
\end{aligned}$$

and finally, using Eqs. (4.15) and (4.17),

$$r_{k+1} = \left(\sum_{i=1}^n \tilde\zeta(i)\phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^n \tilde\zeta(i)\phi(i)D_t(i). \tag{4.18}$$

We can now compare the λ-PI iteration (4.16) and the simulation-based implementation (4.18). Since $\big(T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)\big)(i) = \lim_{t\to\infty} D_t(i)$ and $\zeta(i) = \lim_{t\to\infty}\tilde\zeta(i)$, we see that these two iterations asymptotically coincide.

The expression (4.18) provides some insight on how λ-PI(1) approximates the λ-PI iteration (4.16) [or equivalently $\Phi r_{k+1} = \hat\Pi T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)$; cf. Eq. (4.13)]. Generally the simulation process of λ-PI(1) (many short trajectories) involves more noise than the simulation process of the other implementations (a single long trajectory), because the length of each simulation trajectory is random (exponentially distributed). This can be seen from iteration (4.18), which involves considerable simulation noise due to the presence of $\tilde\zeta(i)$ and $D_t(i)$. However, we will argue that from a practical point of view much of this noise does not play a significant role. To see this, first note that the deviation of $\tilde\zeta(i)$ from $\zeta(i)$, is not important since $\tilde\zeta(i)$ simply redefines the projection norm. Next note that $D_t(i)$ can be written as

$$D_t(i) = \sum_{\ell=0}^\infty \tilde f_\ell(i)\tilde E_\ell(i), \tag{4.19}$$

where $\tilde{f}_\ell(i)$ and $\tilde{E}_\ell(i)$ are the following empirical averages over the entire simulation process:

(a) $\tilde{f}_\ell(i)$ is the empirical relative frequency of cost samples that start at state i, and correspond to trajectories consisting of $\ell+1$ transitions. As $t \to \infty$ it converges to $(1-\lambda)\lambda^\ell$ based on the way the simulation is structured.

(b) $\tilde{E}_\ell(i)$ is the Monte Carlo estimate of the cost of trajectories that start at state i, consist of $\ell+1$ transitions, and have terminal cost vector Φr_k. As $t \to \infty$ it converges to $T^{\ell+1}_{\mu_{k+1}}(\Phi r_k)(i)$.

While both $\tilde{f}_\ell(i)$ and $\tilde{E}_\ell(i)$ contribute to the variance of $D_t(i)$, only $\tilde{E}_\ell(i)$ has practical significance. To see this note that based on Eq. (4.19), $D_t(i)$ can also be viewed as an estimate of

$$\tilde{T}_{\mu_{k+1}}(\Phi r_k)(i) = \sum_{\ell=0}^{\infty} \tilde{f}_\ell(i) T^{\ell+1}_{\mu_{k+1}}(\Phi r_k)(i). \tag{4.20}$$

Thus iteration (4.18) may also be viewed as a simulation-based implementation of the optimistic PI method

$$\Phi r_{k+1} = \tilde{\Pi}\tilde{T}_{\mu_{k+1}}(\Phi r_k),$$

where $\tilde{\Pi}$ is projection with respect to the weighted sup-norm defined by $\tilde{\zeta}$. From a practical point of view, this iteration and the λ-PI iteration $\Phi r_{k+1} = \hat{\Pi} T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)$ perform similarly: there is only a difference in the projection norm ($\tilde{\Pi}$ rather than $\hat{\Pi}$), and a difference in the weights of the terms $T^{\ell+1}_{\mu_{k+1}}$ [$\tilde{f}_\ell(i)$ rather than $(1-\lambda)\lambda^\ell$]; compare $\tilde{T}_{\mu_{k+1}}(\Phi r_k)(i)$ as given by Eq. (4.20) with

$$T^{(\lambda)}_{\mu_{k+1}}(\Phi r_k)(i) = \sum_{\ell=0}^{\infty}(1-\lambda)\lambda^\ell T^{\ell+1}_{\mu_{k+1}}(\Phi r_k)(i),$$

the definition of $T^{(\lambda)}_{\mu_{k+1}}$. Neither difference should affect significantly the quality of the obtained approximation Φr_{k+1}.

In conclusion, with the λ-PI(1) implementation (4.10)-(4.12), as $t \to \infty$, we obtain in the limit the λ-PI iteration Eq. (4.13), with comparable performance degradation due to simulation noise as for the LSPE(λ) implementation of Section 4.1. A key characteristic of the implementation is that it deals with the issue of exploration flexibly and effectively. Since a trajectory of the stopping problem is completed at each transition with the potentially large probability $1-\lambda$, a restart with a new initial state i_0 is frequent and the length of each of the simulated trajectories is relatively small. The restart mechanism can be used as a "natural" form of exploration, by choosing appropriately the restart distribution ζ_0 so that $\zeta(i)$ reflects a "substantial" weight for all states i. Thus λ-PI(1) is like LSPE(λ) (Section 4.1), but with built-in exploration enhancement. Compared to λ-PI(0) (Section 4.2) it involves reduced bias since it aims to find the limit point of TD(λ), not TD(0). In particular, as $\lambda \to 1$, it produces an evaluation Φr_{k+1} that tends to the fixed point of TD(1), i.e., the projection $\hat{\Pi} J_{\mu_{k+1}}$.

4.4 Comparison with Alternative Approximate PI Methods

The preceding λ-PI implementations are in direct competition with approximate PI methods that use LSTD(λ) for policy evaluation. A popular method, often referred to as LSPI (Lagoudakis and Parr [LaP03]), can be simply described as approximate PI combined with LSTD(0) for policy evaluation. The LSPI and λ-PI(0) methods have been compared in [ThS10a] in terms of four characteristics.

(a) *Bias*: Both methods are subject to qualitatively similar bias [they aim to find the limit point of TD(0)].

(b) *Sample efficiency*: Both methods can reuse the same set of sample state trajectories over all policies. (In the model-free case where Q-factors are approximated, again the set of sample state-control trajectories is reusable.)

(c) *Exploration*: Both methods provide the same options for exploration, since the validity of these methods does not depend on whether the simulation trajectories are obtained by using the current policy [in fact these trajectories are reusable as per (b) above].

(d) *Optimistic operation*: Since λ-PI(0) has an iterative character (r_{k+1} depends on r_k), it is less susceptible to simulation noise and has an advantage over LSPI in the case where the number of samples per policy is low. Indeed this assertion is made by Thiery and Scherrer [ThS10a] based on experimentation, who also found that the effect of the choice of λ is more pronounced in this case.

Note that (b) and (c) above are the advantages of LSPI and λ-PI(0) over the LSPE(λ) implementation of Section 4.1 (which in turn involves less bias because of the use of $\lambda > 0$, and also has an optimistic character).

Let us now compare λ-PI(1) with LSPI and λ-PI(0) in terms of the characteristics (a)-(d) above. It has better bias characteristics as noted earlier. It has worse sample efficiency as it cannot reuse simulation trajectories (it can only reuse the restart state sequence). It deals with exploration about as well, thanks to the restart mechanism of the SSP formulation. Finally, like λ-PI(0), λ-PI(1) has an optimistic character, and has a similar advantage over LSPI in this regard, cf. (d) above.

4.5 Exploration-Enhanced LSTD(λ) with Geometric Sampling

The geometric sampling idea underlying the λ-PI(1) implementation of Eqs. (4.10)-(4.12) may also be modified to obtain an exploration-enhanced version of LSTD(λ). In particular, we use the same simulation procedure, and in analogy to Eq. (4.10) we define

$$c_{\ell,m}(r) = \alpha^{N_m-\ell}\phi(i_{N_m,m})'r + \sum_{q=\ell}^{N_m-1} \alpha^{q-\ell} g(i_{q,m}, u_{q,m}, i_{q+1,m}).$$

We then obtain an approximation $\Phi\hat r$ to the solution of the projected equation

$$\Phi r = \hat\Pi T^{(\lambda)}_{\mu_{k+1}}(\Phi r),$$

[cf. Eq. (4.13)] by finding $\hat r$ such that

$$\hat r = \arg\min_{r\in\Re^s} \sum_{m=1}^{t}\sum_{\ell=0}^{N_m-1} \big(\phi(i_{\ell,m})'r - c_{\ell,m}(\hat r)\big)^2. \tag{4.21}$$

By writing the optimality condition

$$\sum_{m=1}^{t}\sum_{\ell=0}^{N_m-1} \phi(i_{\ell,m})\big(\phi(i_{\ell,m})'\hat r - c_{\ell,m}(\hat r)\big) = 0$$

for the least squares minimization in Eq. (4.21) and solving for $\hat r$, we obtain the following implementation of LSTD(λ):

$$\hat r = \hat C^{-1}\hat d, \tag{4.22}$$

where

$$\hat{C} = \sum_{m=1}^{t} \sum_{\ell=0}^{N_m-1} \phi(i_{\ell,m})\big(\phi(i_{\ell,m}) - \alpha^{N_m-\ell}\phi(i_{N_m,m})\big)', \tag{4.23}$$

and

$$\hat{d} = \sum_{m=1}^{t} \sum_{\ell=0}^{N_m-1} \phi(i_{\ell,m}) \sum_{q=\ell}^{N_m-1} \alpha^{q-\ell} g(i_{q,m}, u_{q,m}, i_{q+1,m}). \tag{4.24}$$

For a large number of trajectories t, the exploration-enhanced LSTD(λ) method (4.21) [or equivalently (4.22)-(4.24)] and λ-PI(1) [cf. Eq. (4.12)] yield similar results, particularly when $\lambda \approx 1$. However, λ-PI(1) has an iterative character (r_{k+1} depends on r_k), so it is reasonable to expect that it is less susceptible to simulation noise in an optimistic PI setting where the number of samples per policy is low.

As an example, when $\lambda = 0$, all the simulation trajectories consist of a single transition, so $N_m = 1$ for all $m = 1, \ldots, t$. Then, using Eqs. (4.23) and (4.24), the equation $\hat{C}r = \hat{d}$ becomes

$$\sum_{m=1}^{t} \phi(i_{0,m})\big(\phi(i_{0,m}) - \alpha\phi(i_{1,m})\big)' r = \sum_{m=1}^{t} \phi(i_{0,m}) g(i_{0,m}, u_{0,m}, i_{1,m}).$$

It yields the same vector $\hat{r} = \hat{C}^{-1}\hat{d}$ as the LSTD(0) method that simulates t independent transitions according to the restart distribution ζ_0, rather than simulating a single long trajectory. In fact this is the policy evaluation process in the LSPI method mentioned in Section 4.4. The geometric sampling procedure described here allows exploration-enhancement for any λ.

5. CONCLUSIONS

We discussed a few implementations of λ-PI with linear cost function approximation, which have different strengths and weaknesses with respect to dealing with the critical issues of bias and exploration. Out of the three implementations, the one of Section 4.3, λ-PI(1), is new and seems capable of dealing well with both issues, although it has worse sample complexity than the λ-PI(0) implementation of Section 4.2.

On the other hand, our discussion has been somewhat speculative, and our assessments, while relying on past computational experience, still require supportive experimentation. Moreover, the λ-PI implementations should be compared to other approximate PI methods based on projected equations, such as the exploration-enhanced LSTD(λ) method for policy evaluation, discussed in Section 3, and the LSPI method discussed in Section 4.4. A computational comparison of λ-PI(0) with this latter method is given in [ThS10a], and a similar comparison with λ-PI(1) would be desirable.

Fundamentally, λ-PI(1) is based on geometric sampling, a new simulation idea for λ-methods that uses multiple short trajectories with exploration-enhanced restart, rather than a single infinitely long trajectory. This idea can also be applied to LSTD(λ), thereby obtaining a new exploration-enhanced version of this method, which has been described in Section 4.5.

6. REFERENCES

[BBD10] Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D., 2010. Reinforcement Learning and Dynamic Programming Using Function Approximators, CRC Press, N. Y.

[BBN04] Bertsekas, D. P., Borkar, V. S., and Nedić, A., 2004. "Improved Temporal Difference Methods with Linear Function Approximation," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, and D. Wunsch (Eds.), IEEE Press, N. Y.

[BSA83] Barto, A. G., Sutton, R. S., and Anderson, C. W., 1983. "Neuronlike Elements that Can Solve Difficult Learning Control Problems," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 13, pp. 835-846.

[BeI96] Bertsekas, D. P., and Ioffe, S., 1996. "Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming," Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT.

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[BeY09] Bertsekas, D. P., and Yu, H., 2009. "Projected Equation Methods for Approximate Solution of Large Linear Systems," Journal of Computational and Applied Mathematics, Vol. 227, pp. 27-50.

[BeY10a] Bertsekas, D. P., and Yu, H., 2010. "Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," Lab. for Information and Decision Systems Report LIDS-P-2831, MIT.

[BeY10b] Bertsekas, D. P., and Yu, H., 2010. "Asynchronous Distributed Policy Iteration in Dynamic Programming," Proc. of Allerton Conf. on Information Sciences and Systems.

[Ber95] Bertsekas, D. P., 1995. "A Counterexample to Temporal Differences Learning," Neural Computation, Vol. 7, pp. 270-279.

[Ber07] Bertsekas, D. P., 2007. Dynamic Programming and Optimal Control, 3rd Edition, Vol. II, Athena Scientific, Belmont, MA.

[Ber09] Bertsekas, D. P., 2009. "Projected Equations, Variational Inequalities, and Temporal Difference Methods," Lab. for Information and Decision Systems Report LIDS-P-2808, MIT.

[Ber10] Bertsekas, D. P., 2010. "Pathologies of Temporal Difference Methods in Approximate Dynamic Programming," Proc. 2010 IEEE Conference on Decision and Control.

[Ber11a] Bertsekas, D. P., 2011. Approximate Dynamic Programming, on-line at
http://web.mit.edu/dimitrib/www/dpchapter.html.

[Ber11b] Bertsekas, D. P., 2011. "Approximate Policy Iteration: A Survey and Some New Methods," J. of Control Theory and Applications, Vol. 9, pp. 310-335.

[Ber11c] Bertsekas, D. P., 2011. "Temporal Difference Methods for General Projected Equations," IEEE Trans. on Aut. Control, Vol. 56, pp. 2128-2139.

[Bor08] Borkar, V. S., 2008. Stochastic Approximation: A Dynamical Systems Viewpoint, Cambridge Univ. Press.

[Bor09] Borkar, V. S., 2009. "Reinforcement Learning: A Bridge Between Numerical Methods and Monte Carlo," in World Scientific Review, Vol. 9, Chapter 4.

[BrB96] Bradtke, S. J., and Barto, A. G., 1996. "Linear Least-Squares Algorithms for Temporal Difference Learning," Machine Learning, Vol. 22, pp. 33-57.

[CFH07] Chang, H. S., Fu, M. C., Hu, J., Marcus, S. I., 2007. Simulation-Based Algorithms for Markov Decision Processes, Springer, N. Y.

[CaR11] Canbolat, P. G., and Rothblum, U. G., 2011. "(Approximate) Iterated Successive Approximations Algorithm for Sequential Decision Processes," Technical Report, The Technion - Israel Institute of Technology, May 2011.

[Cao07] Cao, X. R., 2007. Stochastic Learning and Optimization: A Sensitivity-Based Approach, Springer, N. Y.

[FRF11] Foderaro, G., Raju, V., and Ferrari, S., 2011. "A Model-Based Approximate λ-Policy Iteration Approach to Online Evasive Path Planning and the Video Game Ms. Pac-Man," J. of Control Theory and Applications, Vol. 9, pp. 391-399.

[Fle84] Fletcher, C. A. J., 1984. Computational Galerkin Methods, Springer-Verlag, N. Y.

[Gos03] Gosavi, A., 2003. Simulation-Based Optimization Parametric Optimization Techniques and Reinforcement

Learning, Springer-Verlag, N. Y.

[Hay08] Haykin, S., 2008. Neural Networks and Learning Machines (3rd Edition), Prentice-Hall, Englewood-Cliffs, N. J.

[Kra72] Krasnoselskii, M. A., et. al, 1972. Approximate Solution of Operator Equations, Translated by D. Louvish, Wolters-Noordhoff Pub., Groningen.

[LLL08] Lewis, F. L., Lendaris, G. G., and Liu, D., 2008. Special Issue on Adaptive Dynamic Programming and Reinforcement Learning in Feedback Control, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 38.

[LaP03] Lagoudakis, M. G., and Parr, R., 2003. “Least-Squares Policy Iteration,” J. of Machine Learning Research, Vol. 4, pp. 1107-1149

[LeV09] Lewis, F. L., and Vrabie, D., 2009. “Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control,” IEEE Circuits and Systems Magazine, 3rd Q. Issue.

[Mey07] Meyn, S., 2007. Control Techniques for Complex Networks, Cambridge University Press, N. Y.

[NeB03] Nedić, A., and Bertsekas, D. P., 2003. “Least Squares Policy Evaluation Algorithms with Linear Function Approximation,” Discrete Event Dynamic Systems: Theory and Applications, Vol. 13, pp. 79-110.

[Pow07] Powell, W. B., 2007. Approximate Dynamic Programming: Solving the Curses of Dimensionality, Wiley, N. Y.

[Put94] Puterman, M. L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming, J. Wiley, N. Y.

[Rot79] Rothblum, U. G., 1979. “Iterated Successive Approximation for Sequential Decision Processes,” in Stochastic Control and Optimization, by J. W. B. van Overhagen and H. C. Tijms (eds), Vrije University, Amsterdam.

[SBP04] Si, J., Barto, A., Powell, W., and Wunsch, D., (Eds.) 2004. Learning and Approximate Dynamic Programming, IEEE Press, N. Y.

[Sam59] Samuel, A. L., 1959. “Some Studies in Machine Learning Using the Game of Checkers,” IBM Journal of Research and Development, pp. 210-229.

[Sam67] Samuel, A. L., 1967. “Some Studies in Machine Learning Using the Game of Checkers. II – Recent Progress,” IBM Journal of Research and Development, pp. 601-617.

[Sch10] Scherrer, B., 2010. “Should One Compute the Temporal Difference Fix Point or Minimize the Bellman Residual? The Unified Oblique Projection View,” in ICML’10: Proc. of the 27th Annual International Conf. on Machine Learning.

[Sch11] Scherrer, B., 2011. “Performance Bounds for Lambda Policy Iteration and Application to the Game of Tetris,” Report RR-6348, INRIA.

[SuB98] Sutton, R. S., and Barto, A. G., 1998. Reinforcement Learning, MIT Press, Cambridge, MA.

[Sut88] Sutton, R. S., 1988. “Learning to Predict by the Methods of Temporal Differences,” Machine Learning, Vol. 3, pp. 9-44.

[SzL06] Szita, I., and Lorinz, A., 2006. “Learning Tetris Using the Noisy Cross-Entropy Method,” Neural Computation, Vol. 18, pp. 2936-2941.

[Sze10] Szepesvari, C., 2010. “Reinforcement Learning Algorithms for MDPs,” Morgan and Claypool Publishers.

[ThS09] Thiery, C., and Scherrer, B., 2009. “Improvements on Learning Tetris with Cross-Entropy,” International Computer Games Association Journal, Vol. 32, pp. 23-33.

[ThS10a] Thiery, C., and Scherrer, B., 2010. “Least-Squares Policy Iteration: Bias-Variance Trade-off in Control Problems,” Proc. of 2010 ICML, Haifa, Israel.

[ThS10b] Thiery, C., and Scherrer, B., 2010. “Performance Bound for Approximate Optimistic Policy Iteration,” Technical Report, INRIA.

[TsV97] Tsitsiklis, J. N., and Van Roy, B., 1997. "An Analysis of Temporal-Difference Learning with Function Approximation," IEEE Transactions on Automatic Control, Vol. 42, pp. 674–690.

[WPB09] Wang, M., Polydorides, N., and Bertsekas, D. P., 2009. "Approximate Simulation-Based Solution of Large-Scale Least Squares Problems," Lab. for Information and Decision Systems Report LIDS-P-2819, MIT.

[WaB11a] Wang, M., and Bertsekas, D. P., 2011. "Stabilization of Simulation-Based Iterative Methods for Singular and Nearly Singular Linear Systems," Lab. for Information and Decision Systems Report LIDS-P-2878, MIT.

[WaB11b] Wang, M., and Bertsekas, D. P., 2011. "On the Convergence of Iterative Simulation-Based Methods for Singular Linear Systems," Lab. for Information and Decision Systems Report LIDS-P-2879, MIT.

[Wer09] Werbos, P. J., 2009. "Intelligence in the Brain: A Theory of how it Works and how to Build it," Neural Networks, Vol. 22, pp. 200-212.

[WhS92] White, D., and Sofge, D., 1992. Handbook of Intelligent Control, Van Nostrand Reinhold, N.Y.

[WiB93] Williams, R. J., and Baird, L. C., 1993. "Analysis of Some Incremental Variants of Policy Iteration: First Steps Toward Understanding Actor-Critic Learning Systems," Report NU-CCS-93-11, College of Computer Science, Northeastern University, Boston, MA.

[YuB09] Yu, H., and Bertsekas, D. P., 2009. "Convergence Results for Some Temporal Difference Methods Based on Least Squares," IEEE Trans. on Aut. Control, Vol. 54, 2009, pp. 1515-153.

[YuB10] Yu, H., and Bertsekas, D. P., 2010. "Error Bounds for Approximations from Projected Linear Equations," Mathematics of Operations Research, Vol. 35, pp. 306-329.

[YuB11] Yu, H., and Bertsekas, D. P., 2011. "Q-Learning and Policy Iteration Algorithms for Stochastic Shortest Path Problems," Lab. for Information and Decision Systems Report LIDS-P-2871, MIT.

[Yu10a] Yu, H., 2010. "Least Squares Temporal Difference Methods: An Analysis Under General Conditions," Technical report C-2010-39, Dept. Computer Science, Univ. of Helsinki.

[Yu10b] Yu, H., 2010. "Convergence of Least Squares Temporal Difference Methods Under General Conditions," Proc. of the 27th ICML, Haifa, Israel.