**October 2016 (Revised Jan. 2017 and Nov. 2017)** **Report LIDS-P-3205**

# Proximal Algorithms and Temporal Differences for Large Linear Systems: Extrapolation, Approximation, and Simulation†

**Dimitri P. Bertsekas** ‡

**Abstract**

We consider large linear and nonlinear fixed point problems, and solution with proximal algorithms. We show that there is a close connection between two seemingly different types of methods from distinct fields: 1) Proximal iterations for linear systems of equations, which are prominent in numerical analysis and convex optimization, and 2) Temporal difference (TD) type methods, such as TD($\lambda$), LSTD($\lambda$), and LSPE($\lambda$), which are central in simulation-based approximate dynamic programming/reinforcement learning (DP/RL), and its recent prominent successes in large-scale game contexts, among others.

One benefit of this connection is a new and simple way to accelerate the standard proximal algorithm by extrapolation towards the TD iteration, which generically has a faster convergence rate. Another benefit is the potential integration into the proximal algorithmic context of several new ideas that have emerged in the DP/RL context. We discuss some of the possibilities, and in particular, algorithms that project each proximal iterate onto the subspace spanned by a small number of basis functions, using low-dimensional calculations and simulation. A third benefit is that insights and analysis from proximal algorithms can be brought to bear on the enhancement of TD methods.

The linear fixed point methodology can be extended to nonlinear fixed point problems involving a contraction, thus providing guaranteed and potentially substantial acceleration of the proximal and forward backward splitting algorithms at no extra cost. Moreover, the connection of proximal and TD methods can be extended to nonlinear (nondifferentiable) fixed point problems through new proximal-like algorithms that involve successive linearization, similar to policy iteration in DP.

## 1. INTRODUCTION

In this paper we focus primarily on systems of linear equations of the form

$$x = Ax + b, \tag{1.1}$$

where $A$ is an $n \times n$ matrix and $b$ is a column vector in the $n$-dimensional space $\Re^n$. We denote by $\sigma(M)$ the spectral radius of a square matrix $M$ (maximum over the moduli of the eigenvalues of $M$), and we assume the following.

† This report is an extended version of a paper titled "Proximal Algorithms and Temporal Difference Methods for Solving Fixed Point Problems," which will appear in Computational Optimization and Applications Journal.

‡ The author is with the Dept. of Electr. Engineering and Comp. Science, and the Laboratory for Information and Decision Systems, M.I.T., Cambridge, Mass., 02139.

**Assumption 1.1** The matrix $I - A$ is invertible and $\sigma(A) \leq 1$.

We consider the proximal algorithm, originally proposed for the solution of monotone variational inequalities by Martinet [Mar70] (see also the textbook treatments by Facchinei and Pang [FaP03], Bauschke and Combettes [BaC11], and the author's [Ber15]). This algorithm has the form

$$x_{k+1} = P^{(c)}x_k,$$

where $c$ is a positive scalar, and for a given $x \in \Re^n$, $P^{(c)}x$ denotes the solution of the following equation in the vector $y$:

$$y = Ay + b + \frac{1}{c}(x - y).$$

Under Assumption 1.1, this equation has the unique solution

$$P^{(c)}x = \left(\frac{c+1}{c}I - A\right)^{-1}\left(b + \frac{1}{c}x\right), \tag{1.2}$$

because the matrix $\frac{c+1}{c}I - A$ is invertible, since its eigenvalues lie within the unit circle that is centered at $\frac{c+1}{c}$, so they do not include 0.

When $A$ is symmetric, the system (1.1) is the optimality condition for the minimization

$$\min_{x\in\Re^n}\left\{\frac{1}{2}x'Qx - b'x\right\}, \tag{1.3}$$

where $Q = I - A$ and a prime denotes transposition. The proximal algorithm $x_{k+1} = P^{(c)}x_k$ can then be implemented through the minimization

$$x_{k+1} \in \arg\min_{x\in\Re^n}\left\{\frac{1}{2}x'Qx - b'x + \frac{1}{2c}\|x - x_k\|^2\right\},$$

or

$$x_{k+1} = \left(\frac{1}{c}I + Q\right)^{-1}\left(b + \frac{1}{c}x_k\right).$$

In this case, Assumption 1.1 is equivalent to $Q$ being positive definite, with all eigenvalues in the interval $(0, 2]$. Note, however, that for the minimization problem (1.3), the proximal algorithm is convergent for any positive semidefinite symmetric $Q$, as is well known. Thus Assumption 1.1 is not the most general assumption under which the proximal algorithm can be applied.† Still, however, the assumption covers important types of problems, including the case where $A$ is a contraction with respect to some norm, as well as applications in dynamic programming (DP for short), to be discussed shortly.

Let us denote by $T$ the mapping whose fixed point we wish to find,

$$Tx = Ax + b.$$

We denote by $T^\ell$ the $\ell$-fold composition of $T$, where $\ell$ is a positive integer, and we define addition of a finite number and an infinite number of linear operators in the standard way. We introduce the multistep mapping $T^{(\lambda)}$ given by

$$T^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^\ell T^{\ell+1}, \tag{1.4}$$

† It is possible to scale the eigenvalues of $Q$ to lie in the interval $(0, 2]$ without changing the problem, by multiplying $Q$ and $b$ with a suitable positive scalar. This, however, requires some prior knowledge about the location of the eigenvalues of $Q$.

where $\lambda$ is a scalar with $0 < \lambda < 1$. The series defining $T^{(\lambda)}$ is convergent under Assumption 1.1, as we will discuss later. A principal aim of this paper is to establish the relation between the mappings $T^{(\lambda)}$ and $P^{(c)}$, and the ways in which this relation can be exploited algorithmically to compute its fixed point.

The mapping $T^{(\lambda)}$ has been central in the field that we will refer to as "approximate DP" (the name "reinforcement learning" is also often used in artificial intelligence, and the names "neuro-dynamic programming" and "adaptive dynamic programming" are often used in automatic control, with essentially the same meaning). In particular, $T^{(\lambda)}$ is involved in methods for finding a fixed point of the mapping $\Pi T^{(\lambda)}$, where $\Pi$ is either the identity or some form of projection onto a low-dimensional subspace $S$.† In the DP context, $A$ is a substochastic matrix related to the Markov chain of a policy and the equation $x = Ax + b$ is the Bellman equation for the cost function $x$ of the policy. Equations of the form $x = Tx$ are solved repeatedly within the exact policy iteration method, which generates a sequence of improved cost functions and associated policies. Equations of the form $x = \Pi T^{(\lambda)}x$ are solved within a corresponding approximate policy iteration method. Detailed accounts of the approximate DP context are given in several books, including the ones by Bertsekas and Tsitsiklis [BeT96], Sutton and Barto [SuB98], Si et al. [SBP04], Powell [Pow07], Busoniu et al. [BBD10], Szepesvari [Sze10], Bertsekas [Ber12a], Lewis and Liu [LeL13], and Vrabie, Vamvoudakis, and Lewis [VVL13]. Substantial computational experience has been accumulated with this methodology, and considerable success has been obtained (including prominent achievements with programs that play games, such as Backgammon, Go, and others, at impressive and sometimes above human level; see Tesauro [Tes94], Tesauro and Galperin [TeG96], Scherrer et al. [GMS13], [SMG15], and Silver et al. [MKS15], [SHM16]). In challenging approximate DP applications, the dimension of $A$ is very high, the dimension of the approximation subspace $S$ is low by comparison, and the large-scale computations involved in calculating the fixed point of $\Pi T^{(\lambda)}$ are handled by Monte-Carlo simulation schemes.

A variety of simulation-based methods that involve the mapping $T^{(\lambda)}$, such as TD($\lambda$), LSTD($\lambda$), and LSPE($\lambda$), have been proposed in approximate DP. In particular, the fixed point iteration $x_{k+1} = \Pi T^{(\lambda)}x_k$ (where $\Pi$ is orthogonal projection with respect to a weighted Euclidean norm) has been called PVI($\lambda$) in the author's DP textbook [Ber12a] (PVI stands for Projected Value Iteration). Its simulation-based implementation is the LSPE($\lambda$) method (LSPE stands for Least Squares Policy Evaluation) given in joint works of the author with his collaborators Ioffe, Nedić, Borkar, and Yu [BeI96], [NeB03], [BBN04], [YuB06], [Ber11b]. The simulation-based matrix inversion method that solves the fixed point equation $x = \Pi T^{(\lambda)}x$ is the LSTD($\lambda$) method, given by Bradtke and Barto [BrB96], and further discussed, extended, and analyzed by Boyan [Boy02], Lagoudakis and Parr [LaP03], Nedić and Bertsekas [NeB03], Bertsekas and Yu [BeY09], [YuB12], and Yu [Yu10], [Yu12] (LSTD stands for Least Squares Temporal Differences). TD($\lambda$), proposed by Sutton [Sut88] in the approximate DP setting, is a stochastic approximation method for solving the equation $x = \Pi T^{(\lambda)}x$. It has the form

$$x_{k+1} = x_k + \gamma_k\Big(sample\big(\Pi T^{(\lambda)}x_k\big) - x_k\Big), \tag{1.5}$$

where $sample\big(\Pi T^{(\lambda)}x_k\big)$ is a stochastic simulation-generated sample of $\Pi T^{(\lambda)}x_k$, and $\gamma_k$ is a diminishing

† In approximate DP it is common to replace a fixed point equation of the form $x = F(x)$ with the equation $x = \Pi\big(F(x)\big)$. This approach comes under the general framework of Galerkin approximation, which is widely used in a variety of numerical computation contexts (see e.g., the books by Krasnoselskii [Kra72] and Fletcher [Fle84], and the DP-oriented discussion in the paper by Yu and Bertsekas [YuB10]). A distinguishing feature of approximate DP applications is that $F$ is a linear mapping and the equation $x = \Pi\big(F(x)\big)$ is typically solved by simulation-based methods.

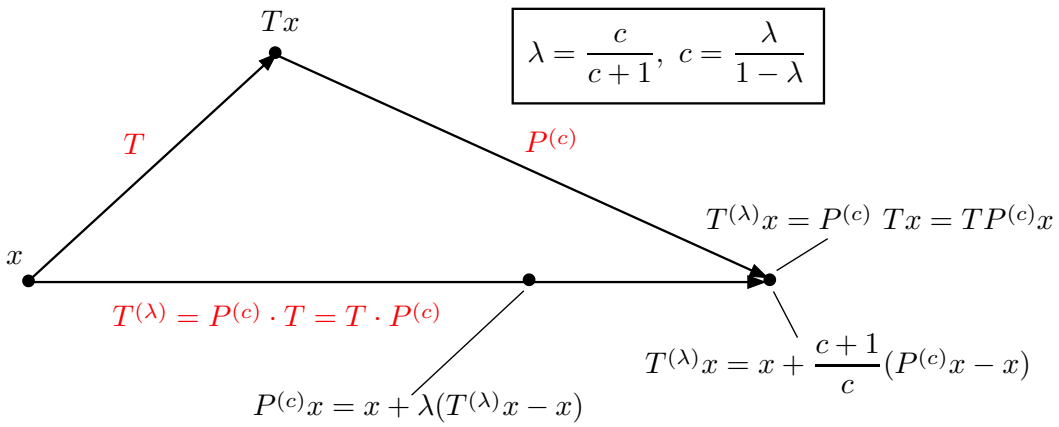


**Figure 1.1.** Relation of the mappings $P^{(c)}$ and $T^{(\lambda)}$. The mapping $P^{(c)}$ is obtained by interpolation between $T^{(\lambda)}$ and the identity. Reversely, $T^{(\lambda)}$ is an extrapolated form of $P^{(c)}$.

stepsize satisfying standard conditions for stochastic iterative algorithms, such as $\gamma_k = 1/(k+1)$.† The computation of the samples in Eq. (1.5) involves simulation using Markov chains and the notion of temporal differences, which originated in reinforcement learning with the works of Samuel [Sam59], [Sam67] on a checkers-playing program. Mathematically, temporal differences are residual-type terms of the form $A^\ell(Ax+b-x)$, $\ell \geq 0$, which can be used to streamline various computations within the aforementioned methods. We refer to the sources given above for methods to generate samples of temporal differences in the DP/policy evaluation context, and to [BeY09] for corresponding methods and analysis within the more general linear fixed point context of the present paper.

A central observation of this paper, shown in Section 2, is that the proximal mapping $P^{(c)}$ is closely related to the multistep mapping $T^{(\lambda)}$, where

$$\lambda = \frac{c}{c+1}.$$

In particular $P^{(c)}$ is an interpolated mapping between $T^{(\lambda)}$ and the identity, or reversely, $T^{(\lambda)}$ is an extrapolated form of $P^{(c)}$; see Fig. 1.1. Moreover, we show in Section 2 that under Assumption 1.1, $T^{(\lambda)}$ has a smaller spectral radius than $P^{(c)}$, and as a result extrapolation of the proximal iterates by a factor $\frac{c+1}{c}$ results in convergence acceleration at negligible computational cost. We also characterize the region of extrapolation factors that lead to acceleration of convergence, and show that it is an interval that contains $(1, 1+1/c]$, but may potentially be substantially larger. These facts are new to the author's knowledge, and they are somewhat unexpected as they do not seem to readily admit an intuitive explanation.

Aside from its conceptual value and its acceleration potential, the relation between $P^{(c)}$ and $T^{(\lambda)}$ suggests the possibility of new algorithmic approaches for large scale applications where the proximal algorithm can be used conveniently. In particular, one may consider the projected proximal algorithm,

$$x_{k+1} = \Pi P^{(c)} x_k,$$

† The precise nature of the problem that TD($\lambda$) is aiming to solve was unclear for a long time. The paper by Tsitsiklis and VanRoy [TsV97] showed that it aims to find a fixed point of $T^{(\lambda)}$ or $\Pi T^{(\lambda)}$, and gave the first convergence analysis (also replicated in the book [BeT96]). The paper by Bertsekas and Yu [BeY09] (Section 5.3) generalized TD($\lambda$), LSTD($\lambda$), and LSPE($\lambda$) to the linear system context of this paper.

which aims to converge to a fixed point of $\Pi P^{(c)}$. The algorithm may be based on simulation-based computations of $\Pi T^{(\lambda)}x$, and such computations have been discussed in the approximate DP context as part of the LSPE($\lambda$) method (noted earlier), and the $\lambda$-policy iteration method (proposed in [BeI96], and further developed in the book [BeT96], and the papers [Ber12b] and [Sch13]). The simulation-based methods for computing $\Pi T^{(\lambda)}x$ have been adapted to the more general linear equation context in [BeY07], [BeY09]; see also [Ber12a], Section 7.3. Another possibility is to use simulation-based matrix inversion to solve the fixed point equation $x = \Pi T^{(\lambda)}x$. In the approximate DP context this is the LSTD($\lambda$) method, which has also been extended to the general linear equations context in [BeY09].

In Section 3 of this paper, we selectively summarize without much analysis how to adapt and transfer algorithms between the TD/approximate DP and the proximal contexts. Our aim is to highlight the algorithmic possibilities that may allow us to benefit from the accumulated implementation experience within these contexts. To this end, we will draw principally on the individual and joint works of the author, M. Wang, and H. Yu; see [BeY07], [BeY09], [YuB10], [Yu10], [Yu12], [Ber11a], [Ber11b], [YuB12], [WaB13], [WaB14], and the textbook account of [Ber12a], Section 7.3, where extensions and analysis of TD($\lambda$), LSTD($\lambda$), and LSPE($\lambda$) for solution of the general linear system $x = \Pi T^{(\lambda)}x$ were given. This includes criteria for $T^{(\lambda)}$ and $\Pi T^{(\lambda)}$ to be a contraction, error bounds, simulation-based implementations, algorithmic variations, dealing with singularity or near singularity of $\Pi P^{(c)}$, etc.

In Section 4 we extend our analysis and algorithms of Section 2 to nonlinear fixed point problems. In particular, we show that an extrapolated form of the proximal algorithm provides increased reduction of the distance to the fixed point over the standard proximal algorithm, provided the fixed point problem has a unique solution and involves a nonexpansive mapping (cf. Assumption 1.1). In Section 4, we also consider forward-backward splitting algorithms and provide a natural generalization of the extrapolation ideas. To our knowledge, these are the first simple extensions of the proximal and forward-backward algorithms for major classes of nonlinear problems, which guarantee acceleration. Other extrapolation methods, such as the ones of [Ber75] and [Ber82], Section 2.3.1 (for convex optimization), or [EcB92] (for monotone operator problems), guarantee convergence but not acceleration, in the absence of additional prior knowledge.

The convergence theory of temporal difference methods is restricted to linear systems that satisfy Assumption 1.1. Thus, for nonlinear fixed point problems, the connection of temporal difference and proximal algorithms seems considerably weaker. To address this situation, we introduce in Section 5 algorithmic ideas based on linearization whereby $T$ is linearized at each iterate $x_k$, and the next iterate $x_{k+1}$ is obtained with a temporal differences-based (exact, approximate, or extrapolated) proximal iteration using the linearized mapping. This approach is similar to Newton's method for solving nonlinear fixed point problems, where the linearized system is solved exactly, rather than approximately (using a single proximal iteration). It is also related to earlier work by Bertsekas and Yu [BeY10], [BeY12], [YuB13] on distributed asynchronous policy iteration algorithms in exact and approximate DP. Let us also mention that minimization algorithms with a composite objective function have been addressed with a prox-linear approach, whereby at each iteration the (differentiable) inner composite function is first linearized and a proximal iteration is applied to the outer composite function; see Lewis and Wright [LeW08], and subsequent works, including the recent papers by Drusvyatskiy and Lewis [DrL16], Duchi and Ryan [DuR17], and the references quoted there. However, these methods deal with minimization rather than fixed point problems and do not seem to be related to the linearized proximal methods of this paper.

## 2. INTERPOLATION AND EXTRAPOLATION FORMULAS

We first review a known result from [BeY09] (Prop. 3) regarding the multistep mapping $T^{(\lambda)}$. By repeatedly applying the formula $x = Ax + b$, we can verify that

$$T^{(\lambda)}x = A^{(\lambda)}x + b^{(\lambda)}, \tag{2.1}$$

where

$$A^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}A^{\ell+1}, \qquad b^{(\lambda)} = \sum_{\ell=0}^{\infty}\lambda^{\ell}A^{\ell}b, \tag{2.2}$$

assuming that the series above are convergent. The following proposition shows that under Assumption 1.1, $T^{(\lambda)}$ is well defined by the power series $(1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}T^{\ell+1}$ [cf. Eq. (1.4)], and that it is a contraction with respect to some norm.

**Proposition 2.1:** Let Assumption 1.1 hold, and let $\lambda \in (0,1)$.

(a) The matrix $A^{(\lambda)}$ and the vector $b^{(\lambda)}$ are well-defined in the sense that the series in Eq. (2.2) are convergent.

(b) The eigenvalues of $A^{(\lambda)}$ have the form

$$\theta_i = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}\zeta_i^{\ell+1} = \frac{\zeta_i(1-\lambda)}{1-\zeta_i\lambda}, \qquad i = 1,\ldots,n, \tag{2.3}$$

where $\zeta_i$, $i = 1,\ldots,n$, are the eigenvalues of $A$. Furthermore, we have $\sigma(A^{(\lambda)}) < 1$ and $\lim_{\lambda\to 1}\sigma\big(A^{(\lambda)}\big) = 0$.

The property $\sigma(A^{(\lambda)}) < 1$ asserted in the preceding proposition, is critical for the subsequent development and depends on the eigenvalues of $A$ being different than 1 (cf. Assumption 1.1). For an intuitive explanation, note that the eigenvalues of $A^{(\lambda)}$ can be viewed as convex combinations of complex numbers from the unit circle at least two of which are different from each other since $\zeta_i \neq 1$ [the nonzero corresponding eigenvalues of $A$ and $A^2$ are different from each other, cf. Eqs. (2.1), (2.3)]. As a result the eigenvalues of $A^{(\lambda)}$ lie within the interior of the unit circle under Assumption 1.1.

The relation between the proximal mapping $P^{(c)}$ and the multistep mapping $T^{(\lambda)}$ is established in the following proposition, which is illustrated in Fig. 1.1.

**Proposition 2.2:** Let Assumption 1.1 hold, and let $c > 0$ and $\lambda = \frac{c}{c+1}$. Then:

(a) $P^{(c)}$ is given by

$$P^{(c)} = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}T^{\ell}, \tag{2.4}$$

and can be written as

$$P^{(c)}x = \overline{A}^{(\lambda)}x + \overline{b}^{(\lambda)}, \qquad x \in \Re^n, \tag{2.5}$$

where

$$\overline{A}^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}A^{\ell}, \qquad \overline{b}^{(\lambda)} = \sum_{\ell=0}^{\infty}\lambda^{\ell+1}A^{\ell}b. \tag{2.6}$$

(b) We have

$$T^{(\lambda)} = TP^{(c)} = P^{(c)}T, \tag{2.7}$$

and for all $x \in \Re^n$,

$$P^{(c)}x = (1-\lambda)x + \lambda T^{(\lambda)}x, \qquad T^{(\lambda)}x = -\frac{1}{c}x + \frac{c+1}{c}P^{(c)}x, \tag{2.8}$$

or equivalently

$$P^{(c)}x = x + \lambda\big(T^{(\lambda)}x - x\big), \qquad T^{(\lambda)}x = x + \frac{c+1}{c}\big(P^{(c)}x - x\big). \tag{2.9}$$

**Proof:** (a) The inverse in the definition of $P^{(c)}$ [cf. Eq. (1.2)] is written as

$$\left(\frac{c+1}{c}I - A\right)^{-1} = \left(\frac{1}{\lambda}I - A\right)^{-1} = \lambda(I-\lambda A)^{-1} = \lambda\sum_{\ell=0}^{\infty}(\lambda A)^\ell,$$

where the power series above is convergent by Prop. 2.1(a). Thus, from Eq. (1.2) and the equation $\frac{1}{c} = \frac{1-\lambda}{\lambda}$,

$$P^{(c)}x = \left(\frac{c+1}{c}I - A\right)^{-1}\left(b + \frac{1}{c}x\right) = \lambda\sum_{\ell=0}^{\infty}(\lambda A)^\ell\left(b + \frac{1-\lambda}{\lambda}x\right) = (1-\lambda)\sum_{\ell=0}^{\infty}(\lambda A)^\ell x + \lambda\sum_{\ell=0}^{\infty}(\lambda A)^\ell b,$$

which from Eq. (2.6), is equal to $\overline{A}^{(\lambda)}x + \overline{b}^{(\lambda)}$, thus proving Eq. (2.5).

(b) We have, using Eqs. (2.1), (2.2), (2.5) and (2.6),

$$TP^{(c)}x = A\big(\overline{A}^{(\lambda)}x + \overline{b}^{(\lambda)}\big) + b = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^\ell A^{\ell+1}x + \sum_{\ell=0}^{\infty}\lambda^{\ell+1}A^{\ell+1}b + b = A^{(\lambda)}x + b^{(\lambda)} = T^{(\lambda)}x,$$

thus proving the left side of Eq. (2.7). The right side is proved similarly. The interpolation/extrapolation formulas (2.8) and (2.9) follow by a straightforward calculation from Eq. (2.4) and the definition $T^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^\ell T^{\ell+1}$ [cf. Eq. (1.4)]. As an example, the following calculation shows the left side of Eq. (2.9):

$$\begin{aligned}
x + \lambda\big(T^{(\lambda)}x - x\big) &= (1-\lambda)x + \lambda T^{(\lambda)}x \\
&= (1-\lambda)x + \lambda\left((1-\lambda)\sum_{\ell=0}^{\infty}\lambda^\ell A^{\ell+1}x + \sum_{\ell=0}^{\infty}\lambda^\ell A^\ell b\right) \\
&= (1-\lambda)\left(x + \sum_{\ell=1}^{\infty}\lambda^\ell A^\ell x\right) + \sum_{\ell=0}^{\infty}\lambda^{\ell+1}A^\ell b \\
&= \overline{A}^{(\lambda)}x + \overline{b}^{(\lambda)} \\
&= P^{(c)}x.
\end{aligned}$$

**Q.E.D**

We will now use the extrapolation formulas of Prop. 2.2(b) to construct variants of the proximal algorithm with interesting properties. The next proposition establishes the convergence and convergence rate properties of the proximal and multistep iterations, and shows how the proximal iteration can be accelerated by extrapolation or interpolation.

**Proposition 2.3:** Let Assumption 1.1 hold, and let $c > 0$ and $\lambda = \frac{c}{c+1}$. Then the eigenvalues of $\overline{A}^{(\lambda)}$ are

$$\overline{\theta}_i = \frac{1-\lambda}{1-\zeta_i\lambda}, \qquad i = 1, \ldots, n, \tag{2.10}$$

where $\zeta_i$, $i = 1, \ldots, n$, are the eigenvalues of $A$. Moreover, $A^{(\lambda)}$ and $\overline{A}^{(\lambda)}$ have the same eigenvectors. Furthermore, we have

$$\frac{\sigma(A^{(\lambda)})}{\sigma(A)} \leq \sigma(\overline{A}^{(\lambda)}) < 1, \tag{2.11}$$

so $\sigma(A^{(\lambda)}) < \sigma(\overline{A}^{(\lambda)})$ if $\sigma(A) < 1$.

**Proof:** Let $e_i$ be an eigenvector of $A^{(\lambda)}$ corresponding to the eigenvalue $\theta_i$. By using the interpolation formula (2.8) and the eigenvalue formula (2.3) for $\theta_i$, we have

$$\overline{A}^{(\lambda)} e_i = (1-\lambda)e_i + \lambda A^{(\lambda)} e_i = \big((1-\lambda) + \lambda\theta_i\big)e_i = \left((1-\lambda) + \lambda\frac{\zeta_i(1-\lambda)}{1-\zeta_i\lambda}\right) e_i = \frac{1-\lambda}{1-\zeta_i\lambda} e_i.$$

Hence, $\overline{\theta}_i = \frac{1-\lambda}{1-\zeta_i\lambda}$ and $e_i$ are the corresponding eigenvalue and eigenvector of $\overline{A}^{(\lambda)}$, respectively.

The proof that $\sigma(\overline{A}^{(\lambda)}) < 1$, or equivalently that $|\overline{\theta}_i| < 1$ for all $i$, follows from a graphical argument on the complex plane, which is given in the caption of Fig. 2.1. We also have from Eqs. (2.3) and (2.10)

$$|\overline{\theta}_i| = \frac{|\theta_i|}{|\zeta_i|}, \qquad i = 1, \ldots, n,$$

which implies that

$$|\overline{\theta}_i| \geq \frac{|\theta_i|}{\sigma(A)}, \qquad i = 1, \ldots, n.$$

By taking the maximum of both sides over $i$, we obtain the left side of Eq. (2.11). **Q.E.D.**

An interesting conclusion can be drawn from Prop. 2.3 about the convergence and the rate of convergence of the proximal iteration $x_{k+1} = P^{(c)}x_k$ and the multistep iteration $x_{k+1} = T^{(\lambda)}x_k$. Under Assumption 1.1, *both iterations are convergent, but the multistep iteration is faster when $A$ is itself a contraction* (with respect to some norm) and is not slower otherwise; cf. Prop. 2.3(a). In the case where $A$ is not a contraction [$\sigma(A) = 1$] it is possible that $\sigma(A^{(\lambda)}) = \sigma(\overline{A}^{(\lambda)})$ (as an example consider a case where all the eigenvalues $\zeta_i$ have modulus 1).

Even in the case where $\sigma(A^{(\lambda)}) = \sigma(\overline{A}^{(\lambda)})$, however, it is possible to accelerate the proximal iteration by interpolating strictly between $P^{(c)}x_k$ and $T^{(\lambda)}x_k$. This is shown in the next proposition, which establishes the convergence rate properties of the extrapolated proximal iteration, and quantifies the range of extrapolation factors that lead to acceleration.

**Proposition 2.4** Let Assumption 1.1 hold, and let $c > 0$ and $\lambda = \frac{c}{c+1}$. Consider any iteration that extrapolates from $P^{(c)}$ in the direction of $T^{(\lambda)}$, i.e.,

$$x_{k+1} = (1-\gamma)P^{(c)}x_k + \gamma T^{(\lambda)}x_k, \qquad \gamma > 0, \tag{2.12}$$

and write it in matrix-vector form as

$$x_{k+1} = A(\lambda,\gamma)x_k + b(\lambda,\gamma),$$

where $A(\lambda,\gamma)$ is an $n \times n$ matrix and $b(\lambda,\gamma) \in \Re^n$. The eigenvalues of $A(\lambda,\gamma)$ are given by

$$\theta_i(\gamma) = (1-\gamma)\overline{\theta}_i + \gamma\theta_i, \qquad i = 1, \ldots, n, \tag{2.13}$$

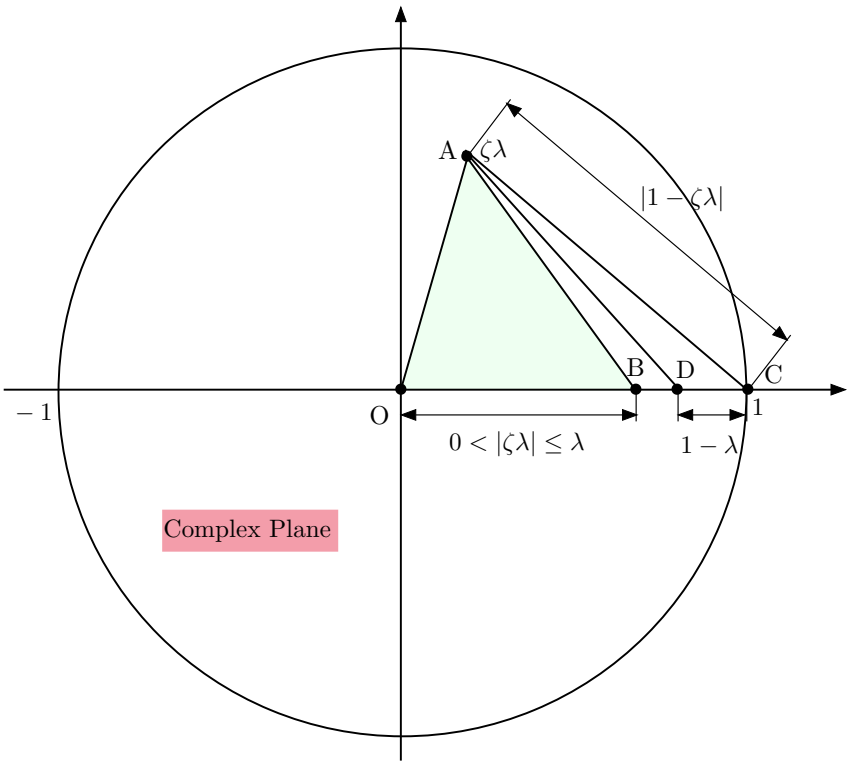


**Figure 2.1.** Proof of the inequality $|\overline{\theta}_i| < 1$, or equivalently that $1 - \lambda < |1 - \zeta\lambda|$ for all complex numbers $\zeta \neq 1$ with $|\zeta| \leq 1$, and $\lambda \in (0, 1)$. We consider the unit circle of the complex plane and the complex number $\zeta\lambda$, and we note that $0 < |\zeta\lambda| \leq \lambda < 1$. If $\zeta$ is in the left-hand side of the plane or on the vertical axis, we clearly have

$$1 - \lambda < 1 \leq |1 - \zeta\lambda|,$$

so it is sufficient to consider the case where $\zeta \neq 0$ and the real part of $\zeta$ is positive, which is depicted in the figure. If $\zeta$ is real, we have $\zeta > 0$ as well as $\lambda > 0$, so

$$|1 - \zeta\lambda| = 1 - \zeta\lambda > 1 - \lambda,$$

and we are done. If $\zeta$ is not real, we consider the isosceles triangle OAB (shaded in the figure), and note that the angles of the triangle bordering the side AB are less than 90 degrees. It follows that the angle ABC and hence also the angle ADC shown in the figure is greater than 90 degrees. Thus the side AC of the triangle ADC is strictly larger than the side DC. This is equivalent to the desired result $1 - \lambda < |1 - \zeta\lambda|$.

and we have

$$\sigma\big(A(\lambda, \gamma)\big) < \sigma(\overline{A}^{(\lambda)}), \tag{2.14}$$

for all $\gamma$ in the interval $(0, \gamma_{\max})$, where

$$\gamma_{\max} = \max\Big\{\gamma > 0 \;\Big|\; \big|\theta_i(\gamma)\big| \leq \overline{\theta}_i, \;\forall\; i = 1, \ldots, n\Big\}.$$

Moreover, we have $\gamma_{\max} \geq 1$, with equality holding if and only if $\sigma(A) = 1$.

**Proof:** The eigenvalue formula (2.13) follows from the interpolation formula

$$A(\lambda, \gamma) = (1 - \gamma)\overline{A}^{(\lambda)} + \gamma A^{(\lambda)},$$

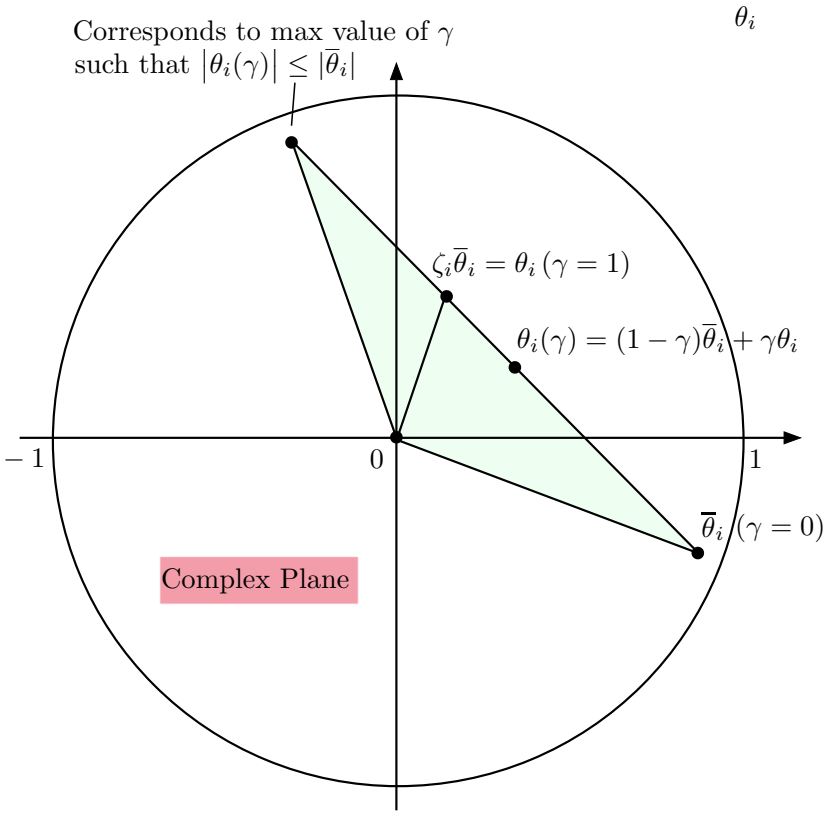


**Figure 2.2.** Illustration of the proof of Prop. 2.4. The eigenvalues $\theta_i(\gamma)$ of $A(\lambda,\gamma)$ are linear combinations (with $\gamma > 0$) of the eigenvalues $\overline{\theta}_i$ and $\theta_i = \zeta_i\overline{\theta}_i$ of $\overline{A}^{(\lambda)}$ and $A^{(\lambda)}$, respectively, and we have $\theta_i \leq \overline{\theta}_i$.

and the fact that $A^{(\lambda)}$ and $\overline{A}^{(\lambda)}$ have the same eigenvectors (cf. Prop. 2.3). For each $i$, the scalar

$$\max\Big\{\gamma > 0 \;\Big|\; |\theta_i(\gamma)| \leq |\overline{\theta}_i|\Big\}$$

is the maximum extrapolation factor $\gamma$ for which $\theta_i(\gamma)$ has at most as large modulus as $\overline{\theta}_i$ (cf. Fig. 2.2), and the inequality (2.14) follows. The inequality $\gamma_{\max} \geq 1$ follows from the construction of Fig. 2.2, since $|\theta_i| \leq |\overline{\theta}_i|$, $\theta_i \neq \overline{\theta}_i$, and $\gamma = 1$ corresponds to the iteration $x_{k+1} = T^{(\lambda)}x_k$. Finally, we have $\gamma_{\max} = 1$ if and only if $|\theta_i| = |\overline{\theta}_i|$ for some $i$, which happens if and only if $|\zeta_i| = 1$ for some $i$, i.e., $\sigma(A) = 1$. **Q.E.D.**

We may implement the extrapolation/interpolation iteration (2.12) by first implementing the proximal iteration $x_{k+1} = P^{(c)}x_k$ and then the multistep iteration according to

$$x_{k+1} = T^{(\lambda)}x_k = x_k + \frac{c+1}{c}\big(P^{(c)}x_k - x_k\big).$$

In this way, unless $\sigma(A) = 1$ [cf. Eq. (2.11)], we achieve acceleration over the proximal iteration. We may then aim for greater acceleration by extrapolating or interpolating between $P^{(c)}x_k$ and $T^{(\lambda)}x_k$ with some factor, possibly determined by experimentation [strict acceleration can always be achieved with $\gamma \in (0,1)$]. This provides a simple and reliable method to accelerate the convergence of the proximal algorithm without

knowledge of the eigenvalue structure of $A$ beyond Assumption 1.1.† Conversely, we may implement the proximal iteration by interpolating the multistep iteration. Some of the possibilities along this direction will be reviewed in the next section.

Finally, let us show that the multistep and proximal iterates $P^{(c)}x_k$ and $T^{(\lambda)}x_k$ can be computed by solving fixed point problems involving a contraction of modulus $\lambda\sigma(A)$.

**Proposition 2.5** Let Assumption 1.1 hold, and let $c > 0$ and $\lambda = \frac{c}{c+1}$. The multistep and proximal iterates $T^{(\lambda)}x_k$ and $P^{(c)}x_k$ are the unique fixed points of the contraction mappings $W_{x_k}$ and $\overline{W}_{x_k}$ given by

$$W_{x_k}x = (1-\lambda)Tx_k + \lambda Tx, \qquad x \in \Re^n,$$

and

$$\overline{W}_{x_k}x = (1-\lambda)x_k + \lambda Tx, \qquad x \in \Re^n,$$

respectively.

**Proof:** Clearly $W_{x_k}$ and $\overline{W}_{x_k}$ are contraction mappings, since they are linear with spectral radius $\lambda\sigma(A) \leq \lambda < 1$. To show that $T^{(\lambda)}x_k$ is the fixed point of $W_{x_k}$, we must verify that $T^{(\lambda)}x_k = W_{x_k}\big(T^{(\lambda)}x_k\big)$, or equivalently that

$$T^{(\lambda)}x_k = (1-\lambda)Tx_k + \lambda T\big(T^{(\lambda)}x_k\big) = (1-\lambda)Tx_k + \lambda T^{(\lambda)}(Tx_k) \tag{2.15}$$

[here we are applying the formula $T\big(T^{(\lambda)}x\big) = T^{(\lambda)}(Tx)$, which is easily verified using Eqs. (2.1) and (2.2)]. In view of the interpolation formula

$$(1-\lambda)x + \lambda T^{(\lambda)}x = P^{(c)}x, \qquad \forall\ x \in \Re^n, \tag{2.16}$$

[cf. Eq. (2.8)], the right-hand side of Eq. (2.15) is equal to $P^{(c)}(Tx_k)$, which from the formula $T^{(\lambda)} = P^{(c)}T$ [cf. Eq. (2.7)], is equal to $T^{(\lambda)}x_k$, the left-hand side of Eq. (2.15).

Similarly, to show that $P^{(c)}x_k$ is the fixed point of $\overline{W}_{x_k}$, we must verify that $P^{(c)}x_k = \overline{W}_{x_k}(P^{(c)}x_k)$, or equivalently that

$$P^{(c)}x_k = (1-\lambda)x_k + \lambda T\big(P^{(c)}x_k\big).$$

This is proved by combining the formula $T^{(\lambda)} = TP^{(c)}$ [cf. Eq. (2.7)], and the interpolation formula (2.16). **Q.E.D.**

The fixed point property of the preceding proposition states that $T^{(\lambda)}x$ is the unique solution of the following equation in $y$:

$$y = (1-\lambda)Tx + \lambda Ty,$$

or

$$y = (1-\lambda)(Ax+b) + \lambda(Ay+b),$$

† It is well known that the proximal iteration can be extrapolated by a factor of as much as two while maintaining convergence. This was first shown for the special case of a convex optimization problem by the author in [Ber75], and for the general case of finding a zero of a monotone operator in Eckstein and Bertsekas [EcB92]; see also a more refined analysis, which quantifies the effects of extrapolation, for the case of a quadratic programming problem, given in the book [Ber82], Section 2.3.1. However, we are not aware of any earlier proposal of a simple and general scheme to choose an extrapolation factor that maintains convergence *and* also guarantees acceleration. Moreover, this extrapolation factor, $(c+1)/c$, may be much larger than two.

and thus provides an explicit matrix inversion formula for the multistep mapping $T^{(\lambda)}$:

$$T^{(\lambda)}x = (1-\lambda A)^{-1}\big(b + (1-\lambda)Ax\big). \tag{2.17}$$

This formula should be compared with the formula (1.2) for the proximal mapping, which can be written in terms of $\lambda$ as

$$P^{(c)}x = (1-\lambda A)^{-1}\big(\lambda b + (1-\lambda)x\big).$$

The fact that the multistep iterate $x_{k+1} = T^{(\lambda)}x_k$ is the fixed point of $W_{x_k}$ is known in exact and approximate DP, and forms the basis for the $\lambda$-policy iteration method; see [BeI96], [BeT96], Section 2.3.1. This is a variant of policy iteration where policy evaluation is done by performing a single multistep iteration using the mapping $T^{(\lambda)}$, where $A$ corresponds to the policy being evaluated. In fact the formula (2.17) is given in [Ber18], Section 4.3.3. In view of our analysis in this paper, it follows that $\lambda$-policy iteration is the approximate version of exact policy iteration, where the exact policy evaluation phase of the latter (which is to find the fixed point of $T$), is approximated with a single (extrapolated) iteration of the proximal algorithm. The $\lambda$-policy iteration method admits some interesting simulation-based implementations, which have been discussed in the approximate DP literature ([Ber12b], [Sch13]), but will not be discussed further here. Based on Prop. 2.5, the proximal iteration $x_{k+1} = P^{(c)}x_k$ admits similar implementations.

Proposition 2.5 also suggests the iteration

$$x_{k+1} = V_m x_k, \tag{2.18}$$

where $V_m x_k$ is obtained by $m > 1$ iterations of the mapping $W_{x_k}$ starting with $x_k$, i.e.,

$$V_m x_k = (W_{x_k})^m x_k,$$

so $V_m x_k$ is an approximate evaluation of $T^{(\lambda)}x_k$, the fixed point of $W_{x_k}$. It can be verified by induction that

$$V_m x_k = (1-\lambda)(Tx_k + \lambda T^2 x_k + \cdots + \lambda^{m-1}T^m x_k) + \lambda^m T^m x_k,$$

and that $V_m$ is a contraction mapping [the preceding formula is given in [BeT96], Prop. 2.7(b), while the contraction property of $V_m$ is proved similar to Prop. 3(a) of [BeY09]]. There is also the similar iteration $x_{k+1} = \overline{V}_m x_k$, where $\overline{V}_m x_k$ is obtained by $m > 1$ iterations of the mapping $\overline{W}_{x_k}$ starting with $x_k$. This iteration may be viewed as an iterative approximate implementation of the proximal algorithm that does not require matrix inversion.

## 3. PROJECTED PROXIMAL, PROXIMAL PROJECTED, AND TEMPORAL DIFFERENCE ALGORITHMS

In this section we aim to highlight situations where analysis and experience from approximate DP can be fruitfully transferred to the solution of linear equations by proximal-like algorithms. In particular, we discuss the simulation-based approximate solution of the equation $x = Tx$ within a subspace $S$ spanned by a relatively small number of basis functions $\phi_1, \ldots, \phi_s \in \Re^n$. Note that while in DP the matrix $A$ is assumed substochastic, the methodology described in this section assumes only Assumption 1.1. The extension beyond the DP framework was first given in [BeY09] and described in further detail in Section 7.3 of the textbook [Ber12a] under the name "Monte-Carlo Linear Algebra."

We denote by $\Phi$ the $n \times s$ matrix whose columns are $\phi_1, \ldots, \phi_s$, so $S$ can be represented as

$$S = \{\Phi r \mid r \in \Re^s\}.$$

Instead of solving $x = Tx$ or the multistep system $x = T^{(\lambda)}x$ (which has the same solution as $x = Tx$) we consider the projected form

$$x = \Pi T^{(\lambda)}x,$$

where the mapping $\Pi : \Re^n \mapsto \Re^n$ is some form of projection onto $S$, in the sense that $\Pi$ is linear, $\Pi x \in S$ for all $x \in \Re^n$, and $\Pi x = x$ for all $x \in S$.

A general form of such $\Pi$ is the oblique projection

$$\Pi = \Phi(\Psi'\Xi\Phi)^{-1}\Psi'\Xi, \tag{3.1}$$

where $\Xi$ is a diagonal $n \times n$ positive semidefinite matrix with components $\xi_1, \ldots, \xi_n$ along the diagonal, and $\Psi$ is an $n \times s$ matrix such that $\Psi'\Xi\Phi$ is invertible. Most often in approximate DP applications, the projection is orthogonal with respect to a Euclidean norm [cf. case (1)] below. However, there are situations where a more general type of projection is interesting [see cases (2) and (3) below]. The use of oblique (as opposed to orthogonal) projections in approximate DP was suggested by Scherrer [Sch10]. We note some special cases that have received attention in the approximate DP literature:

(1) *Orthogonal projection*: Here $\Psi = \Phi$ and $\Xi$ is positive definite. Then $\Pi$ is the orthogonal projection onto $S$ with respect to the norm corresponding to $\Xi$, i.e., $\|x\|^2 = \sum_{i=1}^n \xi_i x_i^2$. Solving the projected system $x = \Pi T^{(\lambda)}x$ corresponds to a form of Galerkin approximation, as noted earlier.†

(2) *Seminorm projection*: Here $\Psi = \Phi$, the matrix $\Phi'\Xi\Phi$ is invertible, but $\Xi$ is only positive semidefinite (so some of the components $\xi_i$ may be zero). Then $\Pi$ is a seminorm projection with respect to the seminorm defined by $\Xi$. The seminorm projection $\Pi x$ can be computed as the unique solution of the least squares problem

$$\min_{r \in \Re^s} \sum_{i=1}^n \xi_i (x_i - \phi_i' r)^2, \tag{3.2}$$

where $\phi_1', \ldots, \phi_n'$ are the rows of $\Phi$ (the solution is unique in view of the assumed invertibility of $\Phi'\Xi\Phi$). This type of least squares problem arises when we are trying to approximate a high dimensional vector $x$ onto $S$ but we know only some of the components of $x$ (but enough to ensure that the preceding minimization has a unique solution). Seminorm projections were first considered in the approximate DP context in the paper by Yu and Bertsekas [YuB12], and we refer to that reference for more discussion on their applications.

(3) *Aggregation*: Here $\Pi = \Phi D$, where $D$ is an $s \times n$ matrix, and we assume that the rows of $\Phi$ and $D$ are probability distributions, and that $\Phi$ and $D$ have a full rank $s$. We replace the solution of the system

† An important fact here is that if $\sigma(A) < 1$ there exists a (weighted) orthogonal projection $\Pi$ such that $\Pi T^{(\lambda)}$ is a contraction for all $\lambda \in (0, 1)$ (see [BeY09] for a more detailed discussion of this issue). While finding such $\Pi$ may not be simple in general, in approximate DP, a suitable projection $\Pi$ is implicitly known in terms of the stationary distribution of a corresponding Markov chain (see e.g., [BeT96], Lemma 6.4, or [Ber12a], Lemma 6.3.1). Another important fact is that if $\sigma(A) \leq 1$, the mapping $\Pi T^{(\lambda)}$ can be made to be a contraction with respect to any norm by taking $\lambda$ sufficiently close to 1 [cf. Prop. 2.1(b)].

$x = Ax + b$ with the projected system $x = \Pi(Ax + b)$, or $\Phi r = \Phi D(A\Phi r + b)$, which equivalently (since $\phi$ has full rank) can be written as

$$r = DA\Phi r + Db.$$

This system is obtained by forming convex combinations of rows and columns of $A$ to construct the "aggregate" matrix $DA\Phi$. Aggregation has a long history in numerical computation, and it is discussed in detail in the context of approximate DP in [Ber11b] and [Ber12a] (Sections 6.5 and 7.3.7), and the references quoted there. It turns out that for a very broad class of aggregation methods, called "aggregation with representative features" ([Ber12a], Example 6.5.4, and Exercise 7.11), the matrix $\Phi D$ is a seminorm projection, as first shown in [YuB12]. The matrix $\Phi D$ can also be viewed as an oblique projection in this case (see [Ber12a], Section 7.3.6).

While the solution of $x = T^{(\lambda)}x$ is the fixed point $x^*$ of $T$ for all $\lambda$, the solution of the projected equation $x = \Pi T^{(\lambda)}x$ depends on $\lambda$. Also, because of the linearity of $\Pi$ and the extrapolation property (2.8) shown in the preceding section, the projected proximal equation $x = \Pi P^{(c)}x$ has the same solution as the system $x = \Pi T^{(\lambda)}x$ where $\lambda = \frac{c}{c+1}$. Let us denote by $x_\lambda$ this solution. Generally, for any norm $\|\cdot\|$ we have the error bound

$$\|x^* - x_\lambda\| \leq \big\|(I - \Pi A^{(\lambda)})^{-1}\big\| \, \|x^* - \Pi x^*\|, \tag{3.3}$$

which is derived from the following calculation:

$$x^* - x_\lambda = x^* - \Pi x^* + \Pi x^* - x_\lambda = x^* - \Pi x^* + \Pi T x^* - \Pi T^{(\lambda)} x_\lambda = x^* - \Pi x^* + \Pi A^{(\lambda)}(x^* - x_\lambda).$$

Thus the approximation error $\|x^* - x_\lambda\|$ is proportional to the "distance" $\big\|x^* - \Pi x^*\big\|$ of the solution $x^*$ from the approximation subspace. It is well known that for values of $\lambda$ close to 1, $x_\lambda$ tends to approximate better the projection $\Pi x^*$. However, values of $\lambda$ close to 1 correspond to large values of $c$, resulting in a less-well conditioned projected proximal equation $x = \Pi P^{(c)}x$. There is also a related tradeoff that is well-documented in the DP literature: as $\lambda$ is increased towards 1, solving the projected equation $x = \Pi T^{(\lambda)}x$ by simulation is more time-consuming because of increased simulation noise and an associated need for more simulation samples. For further discussion of the choice of $\lambda$, we refer to the references cited earlier, and for error bounds that are related but sharper than Eq. (3.3), we refer to Yu and Bertsekas [YuB10], and Scherrer [Sch10], [Sch13].

For the case of the projection formula (3.1) and assuming that $\Phi$ has full rank, the solution $x_\lambda$ of the system

$$x = \Pi T^{(\lambda)}x = \Pi\big(A^{(\lambda)}x + b\big) = \Phi(\Psi'\Xi\Phi)^{-1}\Psi'\Xi\big(A^{(\lambda)}x + b^{(\lambda)}\big),$$

can be written as

$$x_\lambda = \Phi r_\lambda,$$

where $r_\lambda$ is the unique solution of the low-dimensional system of equations

$$r = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi\big(A^{(\lambda)}\Phi r + b^{(\lambda)}\big).$$

Equivalently, this system is written as

$$r = Q^{(\lambda)}r + d^{(\lambda)}, \tag{3.4}$$

where

$$Q^{(\lambda)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi A^{(\lambda)}\Phi, \qquad d^{(\lambda)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi b^{(\lambda)}. \tag{3.5}$$

By defining

$$C^{(\lambda)} = I - Q^{(\lambda)}, \tag{3.6}$$

this system can also be written as

$$C^{(\lambda)} r = d^{(\lambda)}, \tag{3.7}$$

where

$$C^{(\lambda)} = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi \big(I - A^{(\lambda)}\big) \Phi, \qquad d^{(\lambda)} = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi b^{(\lambda)}. \tag{3.8}$$

### 3.1 Projected Proximal and Proximal Projected Algorithms

Let us now consider the solution of the projected equation $x = \Pi T^{(\lambda)} x$, and the equivalent systems $r = Q^{(\lambda)} r + d^{(\lambda)}$ [cf. Eq. (3.4)] and $C^{(\lambda)} r = d^{(\lambda)}$ [cf. Eq. (3.7)] with proximal-type algorithms, assuming that $\Phi$ has full rank. There are two different approaches here (which coincide when there is no projection, i.e., $S = \Re^n$ and $\Pi = I$):

(a) Use the algorithm

$$x_{k+1} = \Pi T^{(\lambda)} x_k, \tag{3.9}$$

or equivalently [cf. Eqs. (3.4) and (3.6)],

$$r_{k+1} = Q^{(\lambda)} r_k + d^{(\lambda)} = r_k - \big(C^{(\lambda)} r_k - d^{(\lambda)}\big). \tag{3.10}$$

Another possibility is to use the interpolated version, which is the projected proximal algorithm

$$x_{k+1} = \Pi P^{(c)} x_k, \tag{3.11}$$

where $c = \frac{\lambda}{1-\lambda}$ (cf. Fig. 3.1), or equivalently, based on the interpolation formula (2.9),

$$r_{k+1} = r_k + \lambda\big(Q^{(\lambda)} r_k + d^{(\lambda)} - r_k\big) = r_k - \lambda\big(C^{(\lambda)} r_k - d^{(\lambda)}\big). \tag{3.12}$$

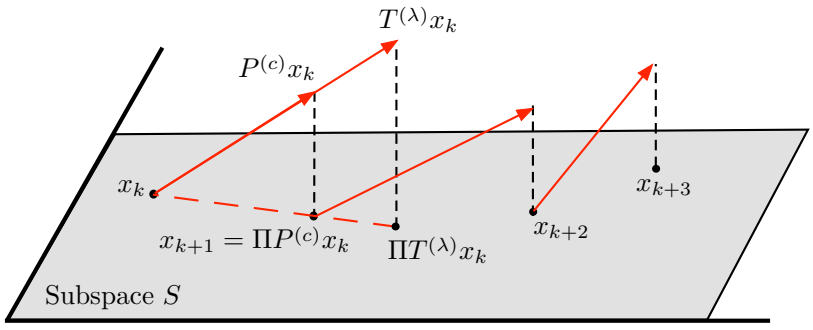


**Figure 3.1.** Illustration of the projected proximal algorithm (3.11) in relation to the projected multistep iteration (3.9). All iterates $x_k$, except possibly $x_0$, lie on $S$.

(b) Apply the proximal algorithm (1.2) to the system $r = Q^{(\lambda)}r + d^{(\lambda)}$ or the system $C^{(\lambda)}r = d^{(\lambda)}$ [cf. Eqs. (3.4) and (3.7)]:

$$\begin{aligned} r_{k+1} &= \left(\frac{\hat{c}+1}{\hat{c}}I - Q^{(\lambda)}\right)^{-1}\left(d^{(\lambda)} + \frac{1}{\hat{c}}r_k\right) \\ &= \left(\frac{1}{\hat{c}}I + C^{(\lambda)}\right)^{-1}\left(d^{(\lambda)} + \frac{1}{\hat{c}}r_k\right) \\ &= r_k - \left(\frac{1}{\hat{c}}I + C^{(\lambda)}\right)^{-1}\left(C^{(\lambda)}r_k - d^{(\lambda)}\right), \end{aligned} \tag{3.13}$$

where $\hat{c}$ is a positive parameter that need not be related to $\lambda$ [cf. Eq. (1.2)]. We may also use the extrapolated version that was discussed in the preceding section:

$$r_{k+1} = r_k - \frac{\hat{c}+1}{\hat{c}}\left(\frac{1}{\hat{c}}I + C^{(\lambda)}\right)^{-1}\left(C^{(\lambda)}r_k - d^{(\lambda)}\right), \tag{3.14}$$

(cf. Fig. 1.1). Extrapolation factors that are intermediate between 1 and $\frac{\hat{c}+1}{\hat{c}}$ or larger than $\frac{\hat{c}+1}{\hat{c}}$ may be considered. Note that this is a *two-parameter algorithm*: the parameter $\hat{c}$ is used for regularization, and may be different from $\frac{1-\lambda}{\lambda}$. This allows some flexibility of implementation: the choice of $\lambda$ should aim to strike a balance between small approximation error $\|x_\lambda - \Pi x^*\|$ and implementation difficulties due to ill-conditioning and/or simulation overhead, while the choice of $\hat{c}$ should aim at guarding against near singularity of $C^{(\lambda)}$. For the extrapolated algorithm (3.14) to be valid, not only should $C^{(\lambda)}$ be invertible, but we must also have $\sigma\big(I - C^{(\lambda)}\big) \leq 1$. This is true under mild conditions; see [BeY09], Prop. 5.

The algorithms in (a) and (b) above are related but different. The algorithms in (a) are *projected proximal* algorithms (possibly with extrapolation), like the ones in Fig. 3.1. The algorithms in (b), use the projection and proximal operations in reverse order: they are *proximal projected* algorithms, i.e., proximal algorithms applied to the projected equation. Both types of algorithms have the generic form

$$r_{k+1} = r_k - \gamma G\big(C^{(\lambda)}r_k - d^{(\lambda)}\big),$$

where $\gamma$ is a nonnegative stepsize and $G$ is a matrix such that $GC^{(\lambda)}$ has eigenvalues with positive real parts. This algorithm and its simulation-based implementations, for both cases where $C^{(\lambda)}$ is nonsingular and singular, has been studied in detail in the papers by Wang and Bertsekas [WaB13] and [WaB14]. Its convergence properties have been analyzed in these references under the assumption that the stepsize $\gamma$ is sufficiently small. The algorithms (3.10) and (3.13) have been explicitly noted in these references. The accelerated proximal projected algorithm (3.14) is new.

Note that the algorithms (3.13) and (3.14) make sense also when $\lambda = 0$, in which case

$$C^{(0)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi(I - A)\Phi, \qquad d^{(0)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi b,$$

[cf. Eq. (3.8)]. Then, the algorithm (3.13) (which is known in approximate DP) can be viewed as the proximal algorithm applied to the projected system $x = \Pi Tx$, while the algorithm (3.14) can be viewed as the corresponding faster multistep algorithm

$$r_{k+1} = r_k - \frac{\hat{c}+1}{\hat{c}}\left(\frac{1}{\hat{c}}I + C^{(0)}\right)^{-1}\left(C^{(0)}r_k - d^{(0)}\right),$$

which has not been considered earlier.

### 3.2 Simulation-Based Methods

The difficulty with the preceding algorithms (3.10)-(3.14) is that when $n$ is very large, the computation of $C^{(\lambda)}$ and $d^{(\lambda)}$ involves high-dimensional inner products, whose exact computation is impossible. This motivates the replacement of $C^{(\lambda)}$ and $d^{(\lambda)}$ with Monte-Carlo simulation-generated estimates, which can be computed with low-dimensional calculations. This simulation-based approach has been used and documented extensively in approximate DP since the late 80s, although the algorithms (3.12) and (3.14) are new, to our knowledge. Moreover, sampling and simulation for solution of linear systems have a long history, starting with a suggestion by von Neumann and Ulam (recounted by Forsythe and Leibler [FoL50]); see also Curtiss [Cur54], [Cur57], and the survey by Halton [Hal70]. More recently, work on simulation methods has focused on using low-order calculations for solving large least squares and other problems. In this connection we note the papers by Strohmer and Vershynin [StV9], Censor, Herman, and Jiang [CeH09], and Leventhal and Lewis [LeL10] on randomized versions of coordinate descent and iterated projection methods for overdetermined least squares problems, and the series of papers by Drineas, Kannan, Mahoney, Muthukrishnan, Boutsidis, and Magdon-Ismail, who consider the use of simulation methods for linear least squares problems and low-rank matrix approximation problems; see [DKM06a], [DKM06b], [DMM06], [DMM08], [DMMS11], and [BDM14].

Let us denote by $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$, respectively, the simulation-based estimates to $C^{(\lambda)}$ and $d^{(\lambda)}$, which are available at iteration $k$. Then the multistep iteration $x_{k+1} = \Pi T^{(\lambda)} x_k$ [cf. Eq. (3.9)] can be implemented in approximate form as

$$r_{k+1} = r_k - \big(C_k^{(\lambda)} r_k - d_k^{(\lambda)}\big), \tag{3.15}$$

[cf. Eq. (3.10)]. We implicitly assume here that at each iteration $k$, one or more Monte-Carlo samples are collected and added to samples from preceding iterations, in order to form $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$, and that the sample collection method is such that

$$\lim_{k\to\infty} C_k^{(\lambda)} = C^{(\lambda)}, \qquad \lim_{k\to\infty} d_k^{(\lambda)} = d^{(\lambda)},$$

with probability 1. The iteration (3.15) is known as LSPE($\lambda$) in approximate DP. Thus, based on the analysis of this paper, LSPE($\lambda$) can be viewed as an extrapolated form of the projected proximal algorithm of Fig. 3.1, implemented by simulation.

A popular alternative to LSPE($\lambda$) is the LSTD($\lambda$) algorithm, which approximates the solution of the projected equation $C^{(\lambda)} r = d^{(\lambda)}$ with the solution of the equation

$$C_k^{(\lambda)} r = d_k^{(\lambda)} \tag{3.16}$$

that iteration (3.15) aims to solve. In LSTD($\lambda$) this is done by simple matrix inversion, but the main computational burden is the calculation of $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$. Analysis and simulation suggests that overall the LSPE($\lambda$) and LSTD($\lambda$) algorithms are computationally competitive with each other; see the discussions in [BBN04], [YuB06], [Ber11b], [Ber12a]. Another prominent algorithm in approximate DP is TD($\lambda$) [cf. Eq. (1.5)], which may be viewed as a stochastic approximation method for solving $C^{(\lambda)} r = d^{(\lambda)}$. This algorithm has a long history in approximate DP, as noted earlier, and has been extended to the general linear system context in [BeY09], Section 5.3.

A major problem for the preceding simulation-based algorithms is that when $C^{(\lambda)}$ is nearly singular, $C_k^{(\lambda)}$ should be a very accurate estimate of $C^{(\lambda)}$, so that $\sigma\big(I - C_k^{(\lambda)}\big) < 1$, which is a requirement for

the methods (3.16) and (3.10) to make sense. The papers [WaB13] and [WaB14] address this issue and provide stabilization schemes to improve the performance of the LSPE($\lambda$) and LSTD($\lambda$) methods, as well as other iterative algorithms for solving singular and near singular systems of equations by simulation. On the other hand the proximal implementations (3.13) and (3.14) are more tolerant of near-singularity of $C^{(\lambda)}$ and simulation noise. This is a generic property of the proximal algorithm and the main motivation for its use. The simulation-based version of the algorithm (3.14), is

$$r_{k+1} = r_k - \frac{\hat{c}+1}{\hat{c}} \left( \frac{1}{\hat{c}} I + C_k^{(\lambda)} \right)^{-1} \left( C_k^{(\lambda)} r_k - d_k^{(\lambda)} \right).$$

Its use of the extrapolation factor $\frac{\hat{c}+1}{\hat{c}}$ may provide significant acceleration, particularly for small values of $\hat{c}$, while simultaneously guarding against near singularity of $C^{(\lambda)}$.

Let us also mention another approach of the proximal projected type, which can also be implemented by simulation. This is to convert the projected equation $C^{(\lambda)} r = d^{(\lambda)}$ to the equivalent equation

$$C^{(\lambda)\prime} \Sigma^{-1} C^{(\lambda)} r = C^{(\lambda)\prime} \Sigma^{-1} d^{(\lambda)},$$

where $\Sigma$ is a symmetric positive definite matrix, and then apply the proximal algorithm to its solution. The analog of the simulation-based proximal algorithm (3.13) is

$$r_{k+1} = r_k - \left( \frac{1}{\hat{c}} I + C_k^{(\lambda)\prime} \Sigma^{-1} C_k^{(\lambda)} \right)^{-1} C_k^{(\lambda)\prime} \Sigma^{-1} \left( C_k^{(\lambda)} r_k - d_k^{(\lambda)} \right), \tag{3.17}$$

and its extrapolated version [cf. Eq. (3.14)] is

$$r_{k+1} = r_k - \frac{\hat{c}+1}{\hat{c}} \left( \frac{1}{\hat{c}} I + C_k^{(\lambda)\prime} \Sigma^{-1} C_k^{(\lambda)} \right)^{-1} C_k^{(\lambda)\prime} \Sigma^{-1} \left( C_k^{(\lambda)} r_k - d_k^{(\lambda)} \right).$$

These algorithms are valid assuming that $C^{(\lambda)}$ is invertible, $\sigma\big(I - C^{(\lambda)\prime} \Sigma^{-1} C^{(\lambda)}\big) \leq 1$, and $\lim_{k\to\infty} C_k^{(\lambda)} = C^{(\lambda)}$. The algorithm (3.17) has been considered both in the approximate DP and the more general linear system context in [WaB13], [WaB14]; see also the textbook presentation of [Ber12a], Sections 7.3.8 and 7.3.9. However, remarkably, if $C^{(\lambda)}$ is singular, its iterate sequence $\{r_k\}$ may diverge as shown by Example 9 of the paper [WaB14] (although the residual sequence $\big\{C_k^{(\lambda)} r_k - d_k^{(\lambda)}\big\}$ can be shown to converge to 0 generically; cf. Prop. 9 of [WaB14]).

### 3.3 The Use of Temporal Differences

The preceding discussion has outlined the general ideas of simulation-based methods, but did not address specifics. Some of the most popular implementations are based on the notion of temporal differences, which are residual-like terms of the form

$$d(x,\ell) = A^\ell (Ax + b - x), \qquad x \in \Re^n,\ \ell = 0, 1, \ldots.$$

In particular, it can be shown using the definition (1.4) that

$$T^{(\lambda)} x = x + \sum_{\ell=0}^{\infty} \lambda^\ell d(x,\ell). \tag{3.18}$$

This can be verified with the following calculation

$$\begin{aligned} T^{(\lambda)}x &= \sum_{\ell=0}^{\infty}(1-\lambda)\lambda^{\ell}(A^{\ell+1}x + A^{\ell}b + A^{\ell-1}b + \cdots + b) \\ &= x + (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}\sum_{m=0}^{\ell}(A^{m}b + A^{m+1}x - A^{m}x) \\ &= x + (1-\lambda)\sum_{m=0}^{\infty}\left(\sum_{\ell=m}^{\infty}\lambda^{\ell}\right)(A^{m}b + A^{m+1}x - A^{m}x) \\ &= x + \sum_{m=0}^{\infty}\lambda^{m}(A^{m}b + A^{m+1}x - A^{m}x) \end{aligned}$$

from [BeY09], Section 5.2.

Based on the least squares implementation (3.2) and the temporal differences expression (3.18), and assuming that $\Phi$ has full rank $s$, the projected algorithm

$$\Phi r_{k+1} = \Pi T^{(\lambda)}\Phi r_k,$$

[cf. the LSPE($\lambda$) algorithm] is given by

$$r_{k+1} = \arg\min_{r\in\Re^s}\sum_{i=1}^{n}\xi_i\left(\phi_i'r - \phi_i'r_k - \sum_{\ell=0}^{\infty}\lambda^{\ell}d_i(\Phi r_k,\ell)\right),$$

where $(\xi_1,\ldots,\xi_n)$ is a probability distribution over the indices $1,\ldots,n$, $\phi_1',\ldots,\phi_n'$ are the rows of $\Phi$, and $d_i(\Phi r_k,\ell)$ is the $i$th component of the $n$-dimensional vector $d(\Phi r_k,\ell)$. Equivalently, this algorithm is written as

$$r_{k+1} = r_k + \left(\sum_{i=1}^{n}\xi_i\phi_i\phi_i'\right)^{-1}\sum_{i=1}^{n}\xi_i\phi_i\left(\sum_{\ell=0}^{\infty}\lambda^{\ell}d_i(\Phi r_k,\ell)\right). \tag{3.19}$$

In the simulation-based implementation of this iteration, the terms

$$\sum_{i=1}^{n}\xi_i\phi_i\phi_i'$$

and

$$\sum_{i=1}^{n}\xi_i\phi_i\left(\sum_{\ell=0}^{\infty}\lambda^{\ell}d_i(\Phi r_k,\ell)\right)$$

are viewed as expected values with respect to the probability distribution $(\xi_1,\ldots,\xi_n)$, and are approximated by sample averages.

The samples may be collected in a variety of ways. Typically, they are obtained by simulating a suitable $n$-state Markov chain to produce one infinite sequence of indexes $(i_0,i_1,\ldots)$ or multiple finite sequences of indexes $(i_0,i_1,\ldots,i_m)$, where $m$ is an integer ($m$ may be different for different sequences). Corresponding samples of the temporal differences are also collected during this process. The transition probabilities of the Markov chain are related to the elements of the matrix $A$, and are chosen in a way to preserve convergence of the iteration (3.19). The details of this, as well as the convergence analysis are complicated, and further review is beyond the scope of this paper. While the formalism of temporal differences is commonly used in practice, simulation-based algorithms may be implemented in other ways (see, e.g., [BeY09], Section 5.1, [Ber12b], [YuB12]). Moreover, similar ideas can be used in simulation-based versions of other related multistep algorithms, such as the one of Eq. (2.18) as well as analogs of the LSTD($\lambda$) and TD($\lambda$) methods. We refer to the literature cited earlier for more detailed discussions.

## 4. EXTENSIONS TO NONLINEAR FIXED POINT PROBLEMS

In this section we consider the solution of the fixed point problem

$$x = T(x), \tag{4.1}$$

where $T : \Re^n \mapsto \Re^n$ is a possibly nonlinear mapping.† The proximal algorithm for this problem has the form

$$x_{k+1} = P^{(c)}(x_k), \tag{4.2}$$

where $c$ is a positive scalar, and for a given $x \in \Re^n$, $P^{(c)}(x)$ solves the following equation in the vector $y$:

$$y = T(y) + \frac{1}{c}(x - y). \tag{4.3}$$

We will operate under assumptions guaranteeing that this equation has a unique solution, so that $P^{(c)}$ will be well defined as a point-to-point mapping.

We focus on the following extrapolated version of the proximal algorithm (4.2):

$$x_{k+1} = E^{(c)}(x_k), \tag{4.4}$$

where

$$E^{(c)}(x) = x + \frac{c+1}{c}\big(P^{(c)}(x) - x\big). \tag{4.5}$$

When $T$ is linear as in Section 1, this algorithm coincides with the multistep method $x_{k+1} = T^{(\lambda)}x_k$ (cf. Fig. 1.1).

The key fact for our purposes is that

$$E^{(c)}(x) = T\big(P^{(c)}(x)\big), \qquad \forall\ x \in \Re^n; \tag{4.6}$$

see Fig. 4.1. Indeed from Eq. (4.3) we have

$$P^{(c)}(x) + \frac{1}{c}\big(P^{(c)}(x) - x\big) = T\big(P^{(c)}(x)\big).$$

Using the form (4.5) of $E^{(c)}(x)$ and the preceding equation, we obtain

$$E^{(c)}(x) = x + \frac{c+1}{c}\big(P^{(c)}(x) - x\big) = P^{(c)}(x) + \frac{1}{c}\big(P^{(c)}(x) - x\big) = T\big(P^{(c)}(x)\big),$$

showing Eq. (4.6).

The form of Eq. (4.6) suggests that the extrapolated iteration (4.4) has faster convergence than the proximal iteration (4.2), within contexts where $T$ is contractive with respect to a suitable norm. In particular, if the solution $P^{(c)}(x)$ of Eq. (4.3) exists and is unique for all $x \in \Re^n$, and $P^{(c)}$ and $T$ are contractions with respect to the same norm, then both iterations (4.2) and (4.4) converge to the unique fixed point of $T$, and the extrapolated iteration converges faster. The following proposition provides specific conditions guaranteeing that this is so.

† For mappings $T$ that may be nonlinear, we use the notation $T(x)$ rather than $Tx$, which is used only when $T$ is known to be linear.

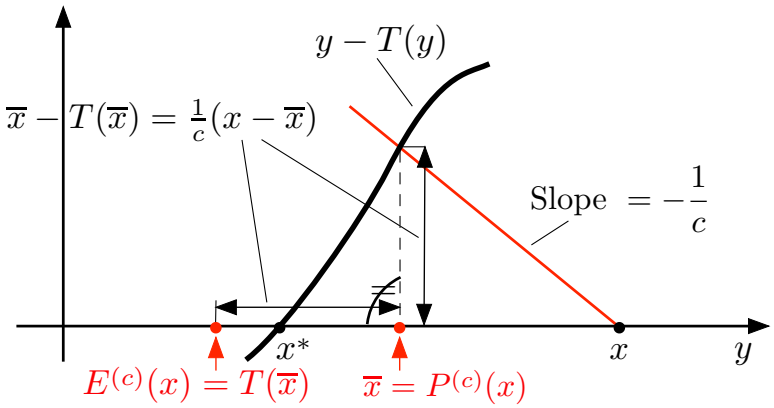


**Figure 4.1.** Illustration of the extrapolated algorithm (4.4). The proximal iterate $P^{(c)}(x)$, denoted $\overline{x}$ in the figure, is extrapolated by $\frac{1}{c}(\overline{x} - x)$. From the definition of $P^{(c)}(x)$, the extrapolated iterate is equal to $T(\overline{x})$ [cf. Eq. (4.6)], and its distance to $x^*$ is strictly smaller than the distance of $\overline{x}$ when $T$ is a contraction.

**Proposition 4.1:** Assume that $T$ is a contraction mapping with modulus $\gamma \in (0,1)$ with respect to a Euclidean norm $\|\cdot\|$, i.e.,

$$\big\|T(x_1) - T(x_2)\big\| \leq \gamma \|x_1 - x_2\|, \qquad \forall\ x_1, x_2 \in \Re^n. \tag{4.7}$$

Then the solution $P^{(c)}(x)$ of Eq. (4.3) exists and is unique for all $x \in \Re^n$, and the mappings $P^{(c)}$ and $E^{(c)}$ are contraction mappings with respect to $\|\cdot\|$. In particular, we have

$$\big\|P^{(c)}(x_1) - P^{(c)}(x_2)\big\| \leq \frac{1}{1 + c(1-\gamma)} \|x_1 - x_2\|, \qquad \forall\ x_1, x_2 \in \Re^n,$$

$$\big\|E^{(c)}(x_1) - E^{(c)}(x_2)\big\| \leq \frac{\gamma}{1 + c(1-\gamma)} \|x_1 - x_2\|, \qquad \forall\ x_1, x_2 \in \Re^n.$$

Moreover, every sequence $\{x_k\}$ generated by either the proximal algorithm (4.2) or its extrapolated version (4.4) converges geometrically to the unique fixed point $x^*$ of $T$, and the convergence of the extrapolated version is faster in the sense that

$$\big\|E^{(c)}(x) - x^*\big\| \leq \gamma \big\|P^{(c)}(x) - x^*\big\|, \qquad \forall\ x \in \Re^n. \tag{4.8}$$

**Proof:** We first verify that the mapping $x \mapsto x - T(x)$ satisfies the standard strong monotonicity assumption under which the proximal mapping is a contraction. In particular, denoting by $\langle \cdot, \cdot \rangle$ the inner product that defines the Euclidean norm $\|\cdot\|$, and using the Cauchy-Schwarz inequality and Eq. (4.7), we have

$$\begin{aligned}
\langle x_1 - x_2, x_1 - T(x_1) - x_2 + T(x_2) \rangle &= \|x_1 - x_2\|^2 - \langle x_1 - x_2, T(x_1) - T(x_2) \rangle \\
&\geq \|x_1 - x_2\|^2 - \|x_1 - x_2\| \cdot \big\|T(x_1) - T(x_2)\big\| \\
&\geq \|x_1 - x_2\|^2 - \gamma \|x_1 - x_2\|^2 \\
&= (1 - \gamma) \|x_1 - x_2\|^2,
\end{aligned} \qquad \forall\ x_1, x_2 \in \Re^n.$$

This relation shows that the mapping $x \mapsto x - T(x)$ is strongly monotone and from standard results, $P^{(c)}$ is well-defined as a point-to-point mapping and we have

$$\big\|P^{(c)}(x_1) - P^{(c)}(x_2)\big\| \leq \frac{1}{1 + c(1-\gamma)} \|x_1 - x_2\|, \qquad \forall\ x_1, x_2 \in \Re^n,$$

(see [Roc76] or [Ber15], Exercise 5.2). In view of Eq. (4.6) and the contraction property of $T$, the corresponding contraction property of $E^{(c)}$ and Eq. (4.8) follow. **Q.E.D.**

### Extrapolation of the Forward-Backward and Proximal Gradient Algorithms

The forward-backward splitting algorithm applies to the fixed point problem $x = T(x) - H(x)$, where $T$ is a maximally monotone point-to-set mapping in a Euclidean space with inner product $\langle \cdot, \cdot \rangle$ defining a Euclidean norm $\| \cdot \|$, and $H$ is single-valued and strongly monotone, in the sense that for some scalar $\beta > 0$, we have

$$\langle x_1 - x_2, H(x_1) - H(x_2) \rangle \geq \beta \|x_1 - x_2\|^2, \qquad \forall\ x_1, x_2 \in \Re^n.$$

The algorithm has the form

$$x_{k+1} = P^{(\alpha)}\big(x_k - \alpha H(x_k)\big), \qquad \alpha > 0,$$

where $P^{(\alpha)}$ is the proximal mapping corresponding to $T$; see Fig. 4.2. This algorithm was analyzed at various levels of generality, by Lions and Mercier [LiM79], Gabay [Gab83], and Tseng [Tse91]. It has been shown to converge to $x^*$ if $\alpha$ is sufficiently small. For a minimization problem where $H$ is the gradient of a strongly convex function, it becomes the popular proximal gradient algorithm; for recent surveys, see Beck and Teboulle [BeT10], Parikh and Boyd [PaB13], and the author's textbook [Ber15] (Ch. 6), among others.

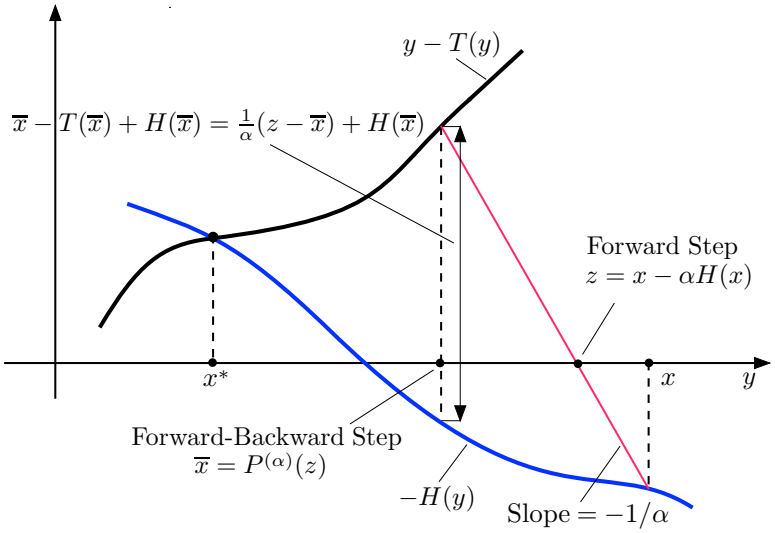


**Figure 4.2.** Illustration of an iteration of the forward backward algorithm.

The extrapolated forward-backward algorithm has the form

$$z_k = x_k - \alpha H(x_k), \qquad \overline{x}_k = P^{(\alpha)}(z_k),$$

$$x_{k+1} = \overline{x}_k + \frac{1}{\alpha}(\overline{x}_k - z_k) - H(\overline{x}_k),$$

and is illustrated in Fig. 4.3. It can be seen that

$$x_{k+1} = T(\overline{x}_k) - H(\overline{x}_k)$$

so there is acceleration if the mapping $T - H$ is contractive. In the linear case where $T(x) = Ax + b$, $H(x) = Bx$, where $A$ and $B$ are $n \times n$ matrices, the algorithm can be related to temporal difference methods, and may be implemented using simulation-based techniques.

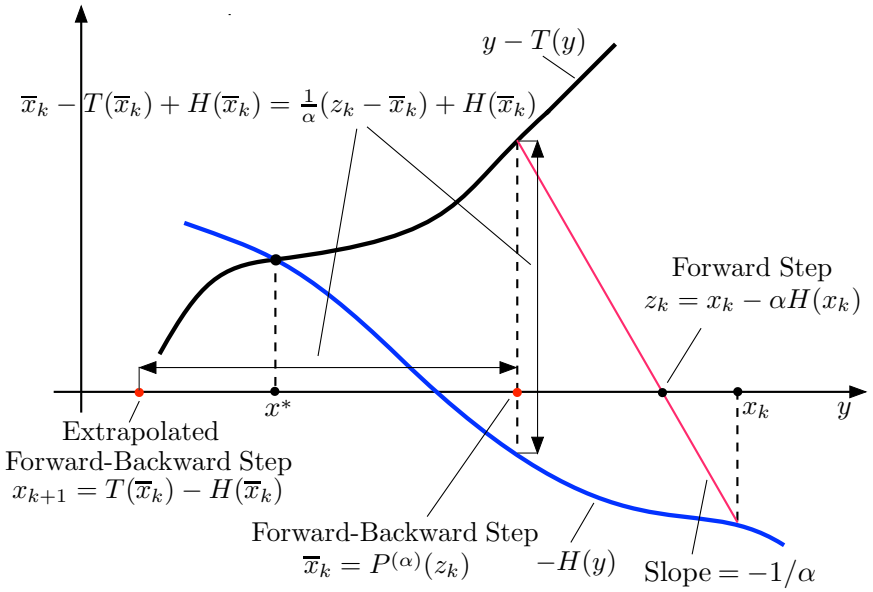


**Figure 4.3.** Illustration of the forward backward algorithm with extrapolation.

## 5. LINEARIZED PROXIMAL AND TEMPORAL DIFFERENCE METHODS FOR NONLINEAR PROBLEMS

The proximal algorithm (4.2) and its extrapolated version (4.5) cannot be related to multistep temporal difference algorithms when $T$ is nonlinear, because then the mapping $P^{(c)}$ does not admit a power series expansion; cf. Eq. (2.4). It is possible, however, to consider algorithmic ideas based on linearization whereby $T$ is linearized at each iterate $x_k$, and the next iterate $x_{k+1}$ is obtained with a temporal differences-based (exact, approximate, or extrapolated) proximal iteration using the linearized mapping. This type of algorithm bears similarity to Newton's method for solving nonlinear fixed point problems, the difference being that the linearized system is solved approximately, using a single proximal iteration, rather than exactly (as in Newton's method). The algorithm does not seem to have been considered earlier, to the author's knowledge, although related ideas underlie the $\lambda$-policy iteration and optimistic policy iteration methods in DP (see the end of Section 2 and the subsequent discussion).

We focus on the fixed point problem $x = T(x)$ with the $i$th component of $T(x)$ having the form

$$T(i,x) = \min_{\mu(i)\in M(i)} \big\{a\big(i,\mu(i)\big)'x + b\big(i,\mu(i)\big)\big\}, \qquad x \in \Re^n,\ i = 1,\ldots,n, \tag{5.1}$$

where for each $i$, $M(i)$ is some set, and $a\big(i,\mu(i)\big)$ and $b\big(i,\mu(i)\big)$ are a (column) vector in $\Re^n$ and scalar, respectively, for each $\mu(i) \in M(i)$. For a given $i$, this form of $T(i,x)$ includes a very broad class of concave functions of $x$. Intuition suggests that an algorithmic analysis for more general forms of $T$ may be possible, but this is beyond the scope of the present paper (see the discussion of Section 5.2).

Let $\mathcal{M}$ be the Cartesian product $M(1) \times \cdots \times M(n)$. Given a vector $\mu = \big((\mu(1),\ldots,\mu(n)\big) \in \mathcal{M}$, the matrix whose $i$th row is the vector $a\big(i,\mu(i)\big)'$ is denoted by $A_\mu$ and the vector whose $i$th component is $b\big(i,\mu(i)\big)$ is denoted by $b_\mu$. We denote by $T_\mu$ the linear mapping given by

$$T_\mu x = A_\mu x + b_\mu,$$

and we write in shorthand

$$T(x) = \min_{\mu \in \mathcal{M}} T_\mu x,$$

where the minimum over $\mu$ above is meant separately, for each of the $n$ components of $T_\mu x$. Our notation here and later is inspired by notation widely used in DP and policy iteration contexts, where $i$ corresponds to state, the components $x(i)$ of $x$ correspond to cost at state $i$, $\mu(i)$ corresponds to control at state $i$, $M(i)$ corresponds to the control constraint set at state $i$, $\mu$ corresponds to policy, $A_\mu$ and $b_\mu$ correspond to the transition probability matrix and cost per stage vector for policy $\mu$, $T_\mu$ is the mapping that defines Bellman's equation for the policy $\mu$, and $T$ is the mapping that defines Bellman's equation for the corresponding Markovian decision problem.

### 5.1 A Linearized Proximal Algorithm Under a Monotonicity Assumption

In this subsection, we will introduce a linearized algorithm, which we will analyze under some assumptions. Chief among these assumptions is that for all $\mu \in \mathcal{M}$, the mapping $T_\mu$ is monotone, in the sense that the matrix $A_\mu$ has nonnegative components, and that the initial condition $x_0$ must satisfy $x_0 \geq T(x_0)$. In the next subsection, we will consider a randomized variant of this algorithm under an alternative set of assumptions.

At the typical iteration, given the current iterate $x_k$, we find $\mu_k(i)$ that attains the minimum over $\mu(i) \in M(i)$ of $a\big(i,\mu(i)\big)'x_k + b\big(i,\mu(i)\big)$, $i = 1,\ldots,n$, and we let $\mu_k = \big(\mu_k(1),\ldots,\mu_k(n)\big)$ (the attainment of the minimum will be assumed in what follows). We denote this by writing

$$T_{\mu_k} x_k = \min_{\mu \in \mathcal{M}} T_\mu x_k = T(x_k),$$

or

$$\mu_k \in \arg\min_{\mu \in \mathcal{M}} T_\mu x_k, \tag{5.2}$$

where the minimum is meant separately, for each of the $n$ components of $T_\mu x_k$. We obtain $x_{k+1}$ via the multistep (extrapolated proximal) iteration

$$x_{k+1} = T_{\mu_k}^{(\lambda)} x_k, \tag{5.3}$$

where for a given $\lambda \in (0,1)$, $T_{\mu_k}^{(\lambda)}$ is the multistep mapping corresponding to the linear mapping $T_{\mu_k}$,

$$T_{\mu_k} x = A_{\mu_k} x + b_{\mu_k}, \qquad x \in \Re^n; \tag{5.4}$$

cf. Eq. (1.4). The algorithm is illustrated in Fig. 5.1, together with its proximal version. Note that $a\big(i,\mu_k(i)\big)$ is the gradient of $T(i,\cdot)$ at $x_k$ if $T(i,\cdot)$ is differentiable, and otherwise it is a subgradient of $T(i,\cdot)$ at $x_k$. This justifies the terms "linearization" and "linearized mapping."

The algorithm (5.3)-(5.4) is related to the $\lambda$-policy iteration method for DP problems where $\{\mu_k\}$ is the sequence of generated policies, and the fixed point equation $x = T_{\mu_k} x$ corresponds to Bellman's equation for the policy $\mu_k$ (see the discussion and references given at the end of Section 2). The algorithm admits several variations where $T_{\mu_k}^{(\lambda)}$ is replaced in Eq. (5.3) by some other related mapping; for example the iteration

$$x_{k+1} = P_{\mu_k}^{(c)} x_k, \tag{5.5}$$

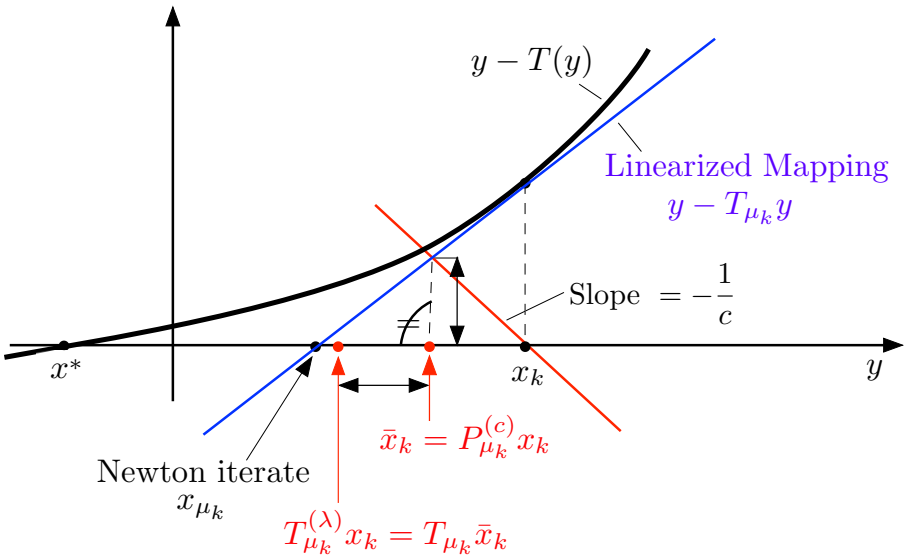


**Figure 5.1.** Illustration of the linearized multistep algorithm (5.2)-(5.3), and its proximal version. At the current iterate $x_k$, we linearize $T$ and find the proximal iterate $\bar{x}_k = P^{(c)}_{\mu_k} x_k$ that aims to find the fixed point $x_{\mu_k}$ of the linearized mapping $T_{\mu_k}$. We can then find the multistep iterate by extrapolation

$$T^{(\lambda)}_{\mu_k} x_k = T_{\mu_k} \bar{x}_k = \bar{x}_k + \frac{1}{c}(\bar{x}_k - x_k);$$

[cf. Eq. (5.3)]. Alternatively, $T^{(\lambda)}_{\mu_k} x_k$ can be found by a temporal differences-based calculation. Note the similarity with the form of Newton's method that finds $x_{\mu_k}$, the unique fixed point of $T_{\mu_k}$, i.e., the iteration $x_{k+1} = x_{\mu_k}$. Newton's method is generally faster but may require much more overhead than the linearized proximal or multistep iteration.

where $P^{(c)}_{\mu_k}$ is the proximal mapping corresponding to $T_{\mu_k}$, or the iteration (2.18). Another related possibility is the iteration

$$x_{k+1} = T^m_{\mu_k} x_k, \tag{5.6}$$

where $T^m_{\mu_k}$ is the composition of $T_{\mu_k}$ with itself $m$ times ($m \geq 1$). This is related to the optimistic policy iteration method of DP; see [BeT96], [Ber12a], or [Ber18], Section 2.5.

Figure 5.2 illustrates the fixed point iterate $T_{\mu_k} x_k$ [Eq. (5.6) with $m = 1$], which is equal to $T(x_k)$. A comparison with Fig. 5.1 shows that the multistep iterate $T^{(\lambda)}_{\mu_k} x_k$ is closer to the Newton iterate $x_{\mu_k}$ than the fixed point iterate $T_{\mu_k} x_k$ for large values of $\lambda$, and in fact converges to $x_{\mu_k}$ as $\lambda \to 1$. The multistep iterate $T^{(\lambda)}_{\mu_k} x_k$ approaches the fixed point iterate $T(x_k) = T_{\mu_k} x_k$ as $\lambda \to 0$. The proximal iterate $P^{(c)}_{\mu_k} x_k$ [cf. Eq. (5.5)] also approaches $x_{\mu_k}$ as $c \to \infty$ (i.e., $\lambda \to 1$), but approaches $x_k$ as $c \to 0$ (i.e., $\lambda \to 0$).

We now introduce a framework for the convergence analysis of the algorithm (5.2)-(5.3). We introduce a set $X \subset \Re^n$, within which we will require that the iterates $x_k$ lie. It is possible that $X = \Re^n$ but some interesting special cases are obtained if $X$ is a strict subset of $\Re^n$, such as for example the nonnegative orthant. Let us say that a vector $\mu \in \mathcal{M}$ is *proper* if $A_\mu$ has eigenvalues strictly within the unit circle and the unique fixed point of $T_\mu$, denoted by $x_\mu$, lies within $X$. If $\mu$ is not proper it is called *improper*. The names "proper" and "improper" relate to notions of proper and improper policies in DP, and stochastic shortest path problems in particular; see [BeT91], [BeT96], [Ber12a], [Ber18]. Note that for proper $\mu$ the algorithmic results of Section 2 come into play, in view of the linearity of $T_\mu$. In particular, the multistep and proximal

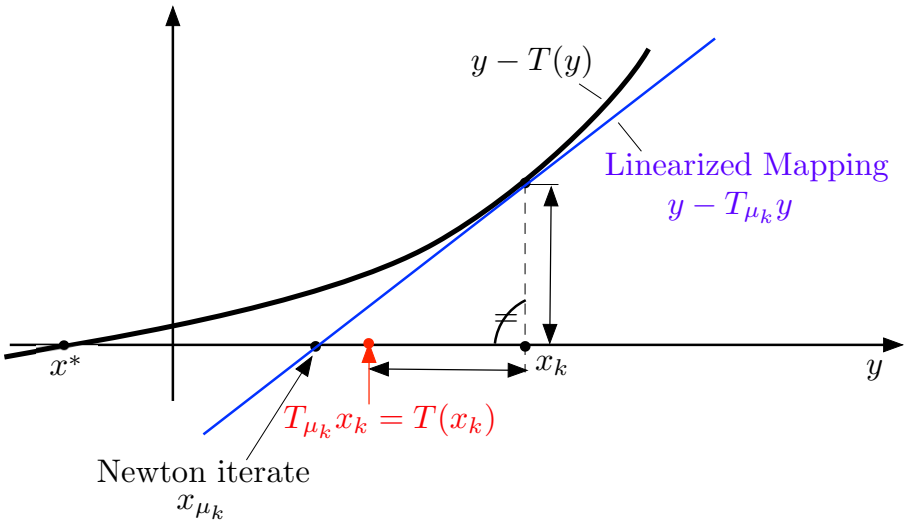


**Figure 5.2.** Illustration of the iterate $T_{\mu_k} x_k$ [Eq. (5.6) with $m = 1$]. It is equal to the fixed point iterate $T(x_k)$.

iterations $x_{k+1} = T_\mu^{(\lambda)} x_k$ and $x_{k+1} = P_\mu^{(c)} x_k$ converge to $x_\mu$ starting from any $x_0 \in \Re^n$.

We will assume the following.

**Assumption 5.1:**

(a) For all $x \in \Re^n$ and $i = 1, \ldots, n$, the minimum over $M(i)$ in Eq. (5.1) is attained.

(b) For all $\mu \in \mathcal{M}$, the matrix $A_\mu$ has nonnegative components.

(c) There exists at least one proper vector, and for each improper vector $\mu$ and $x \in X$, at least one component of the sequence $\{T_\mu^k x\}$ diverges to $+\infty$.

Assumption 5.1(a) is needed to ensure that the algorithm is well defined. Assumption 5.1(b) implies a monotonicity property typically encountered in DP problems, whereby we have for all $\mu \in \mathcal{M}$,

$$T_\mu x \leq T_\mu y \qquad \forall\ x, y \in \Re^n \text{ such that } x \leq y, \tag{5.7}$$

as well as

$$T(x) \leq T(y) \qquad \forall\ x, y \in \Re^n \text{ such that } x \leq y. \tag{5.8}$$

[The relation (5.8) follows from the relation (5.7) by first taking the infimum of the left side to obtain $T(x) \leq T_\mu y$ and then by taking the infimum of the right side over $\mu \in \mathcal{M}$.] The monotonicity property can be replaced by a sup-norm contraction assumption on $A_\mu$, but this requires substantial algorithmic modifications that will be the subject of the next subsection. Note that parts (b) and (c) of Assumption 5.1 are satisfied if $X = \Re^n$, and $A_\mu$ has nonnegative components and eigenvalues strictly within the unit circle for all $\mu \in \mathcal{M}$, in which case all $\mu \in \mathcal{M}$ are proper.

We have the following proposition.

**Proposition 5.1:** Let Assumption 5.1 hold. For a given $\mu \in \mathcal{M}$, if for some $x \in X$ we have $T_\mu x \leq x$, then $\mu$ is proper.

**Proof:** By the monotonicity of $T_\mu$ [cf. Eq. (5.7)], we have $T_\mu^k x \leq x$ for all $k$, so if $\mu$ were improper, Assumption 5.1(c) would be violated. **Q.E.D.**

We now consider a restricted optimization problem over the proper policies, which is inspired from the theory of semicontractive problems in abstract DP (see [Ber18]). We introduce the componentwise minimum vector $x^*$, which has components $x^*(i)$ given by

$$x^*(i) = \inf_{\mu:\text{proper}} x_\mu(i), \qquad i = 1, \ldots, n, \tag{5.9}$$

where $x_\mu(i)$ is the $i$th component of the vector $x_\mu$. The following proposition gives the central results of this subsection.

**Proposition 5.2:** Let Assumption 5.1 hold.

(a) If $T$ has a fixed point within $X$, then this fixed point is equal to $x^*$. Moreover, there exists a proper $\mu$ that attains the infimum in Eq. (5.9).

(b) If $x^* \in X$, then $x^*$ is the unique fixed point of $T$ within $X$.

(c) A sequence $\{x_k\}$ generated by the algorithm (5.2)-(5.3) starting from an initial condition $x_0$ such that $x_0 \geq T(x_0)$ is monotonically nonincreasing and converges to $x^*$. Moreover, in the corresponding sequence $\{\mu_k\}$ all $\mu_k$ are proper.

**Proof:** (a) Let $\hat{x} \in X$ be a fixed point of $T$ within $X$. We will show that $\hat{x} = x^*$. Indeed, using also the monotonicity of $T_\mu$ and $T$ [cf. Eqs. (5.7) and (5.8)], we have for every $m \geq 1$ and proper $\mu$

$$\hat{x} = T(\hat{x}) \leq T_\mu \hat{x} \leq T_\mu^m \hat{x} \leq \lim_{m\to\infty} T_\mu^m \hat{x} = x_\mu,$$

where the second and third inequalities follow from the first inequality and the monotonicity of $T_\mu$. By taking the infimum of the right side over all proper $\mu$, we obtain $\hat{x} \leq x^*$. For the reverse inequality, let $\hat{\mu}$ be such that $\hat{x} = T(\hat{x}) = T_{\hat{\mu}}\hat{x}$ [there exists such $\hat{\mu}$ by Assumption 5.1(a)]. Using Prop. 5.1, it follows that $\hat{\mu}$ is proper, so that $\hat{x}$ is the unique fixed point of $T_{\hat{\mu}}$, i.e., $\hat{x} = x_{\hat{\mu}} \geq x^*$. Thus $\hat{x} = x_{\hat{\mu}} = x^*$ and the proof is complete.

(b) For every proper $\mu$ we have $x_\mu \geq x^*$, so by the monotonicity of $T_\mu$,

$$x_\mu = T_\mu x_\mu \geq T_\mu x^* \geq T x^*.$$

Taking the infimum over all proper $\mu$, we obtain $x^* \geq Tx^*$. Let $\mu$ be such that $Tx^* = T_\mu x^*$. The preceding relations yield $x^* \geq T_\mu x^*$, so by Prop. 5.1, $\mu$ is proper. Therefore, we have

$$x^* \geq T(x^*) = T_\mu x^* \geq \lim_{k\to\infty} T_\mu^k x^* = x_\mu \geq x^*,$$

where the second equality holds since $\mu$ is proper, and $x^* \in X$ by assumption. Hence equality holds throughout in the above relation, which proves that $x^*$ is a fixed point of $T$, which is unique by part (a).

(c) For all $x$ and $\mu \in \mathcal{M}$ such that $x \geq T_\mu x = T(x)$, we claim that

$$x \geq T_\mu x \geq T_\mu^{(\lambda)} x \geq T_\mu \cdot T_\mu^{(\lambda)} x \geq T(T_\mu^{(\lambda)} x), \tag{5.10}$$

To see this, note that the second inequality follows from the power series expansion

$$T_\mu^{(\lambda)} x = (1-\lambda)(T_\mu x + \lambda T_\mu^2 x + \lambda^2 T_\mu^3 x + \cdots), \tag{5.11}$$

and the fact that $x \geq T_\mu x$ implies $T_\mu^m x \geq T_\mu^{m+1} x$ for all $m \geq 0$. The latter fact also implies the third inequality in Eq. (5.10), by applying $T_\mu$ to both sides of Eq. (5.11). We also have

$$T_\mu^{(\lambda)} x \geq x_\mu, \tag{5.12}$$

which follows by taking the limit as $m \to \infty$ in the relation

$$\sum_{\tau=0}^{m} \lambda^t T_\mu^{t+1} x \geq \frac{1-\lambda^{m+1}}{1-\lambda} T_\mu^{m+1} x.$$

Thus from the relations (5.10) and (5.12) we have that if $x \in X$ and $x \geq T(x) = T_\mu x$, then $\mu$ is proper and the vector $\bar{x} = T_\mu^{(\lambda)} x$ satisfies $\bar{x} \in X$, $x \geq \bar{x} \geq T(\bar{x})$, and $x \geq x_\mu \geq x^*$. Applying repeatedly this argument and using the definition of the algorithm we have

$$x_k \geq T(x_k) = T_{\mu_k} x_k \geq T_{\mu_k}^{(\lambda)} x_k = x_{k+1}, \qquad x_k \geq x_{\mu_k} \geq x^*, \qquad k = 0, 1, \ldots. \tag{5.13}$$

It follows that the sequence $\{\mu_k\}$ consists of proper vectors and $x_k \downarrow x_\infty$, where $x_\infty$ is some vector with $x_\infty \geq x^*$.

Next we show that $x_\infty$ is a fixed point of $T$.† Indeed, from the relation $x_k \geq T(x_k)$ [cf. Eq. (5.13)], we have $x_k \geq T(x_\infty)$ for all $k$, so that $x_\infty \geq T(x_\infty)$. Also we note that from the relation $T(x_k) \geq x_{k+1}$ we have

$$\lim_{k\to\infty} T(x_k) \geq x_\infty.$$

Moreover for all $\mu \in \mathcal{M}$, in view of the linearity of $T_\mu$, we have

$$T_\mu x_\infty = \lim_{k\to\infty} T_\mu x_k \geq \lim_{k\to\infty} T(x_k).$$

By combining the preceding two relations, we obtain $T_\mu x_\infty \geq x_\infty$, so by taking the componentwise minimum over $\mu \in \mathcal{M}$, we have $T(x_\infty) \geq x_\infty$. This relation, combined with the relation $x_\infty \geq T(x_\infty)$ shown earlier, proves that $x_\infty$ is a fixed point of $T$.

Finally we show that $x_\infty = x^*$. Indeed, since $x_\infty$ is a fixed point of $T$ and $x^* \leq x_\infty$, as shown earlier, we have

$$x^* \leq x_\infty = T^k(x_\infty) \leq T_\mu^k x_\infty \leq T_\mu^k x_{\mu^0}, \qquad \forall\ \mu : \text{proper},\ k = 0, 1, \ldots.$$

By taking the limit as $k \to \infty$, and using the fact that $x_{\mu_0} \in X$, it follows that $x^* \leq x_\infty \leq x_\mu$ for all $\mu$ proper. By taking the infimum over proper $\mu$, it follows that $x_\infty = x^*$, so $x^*$ is a fixed point of $T$. **Q.E.D.**

The initial condition requirement $x_0 \geq T(x_0)$ is somewhat restrictive and will be removed in the next subsection after the algorithm (5.2)-(5.3) is modified. We can also prove results that are similar to the preceding proposition, but where the nonnegativity Assumption 5.1(b) and the condition $x_0 \geq T(x_0)$ are replaced by alternative conditions. For example, linearization algorithms for finding a fixed point of $T$ are given in the author's monograph [Ber18], Section 2.6.3, under just the assumption that all the matrices $A_\mu$, $\mu \in M$, are contractions with respect to a common weighted sup-norm (see also the papers by Bertsekas and

† Note here that $x_\infty$ may have some components that are equal to $-\infty$. Still, however, because the components of $A_\mu$ are assumed nonnegative, the vectors $T_\mu x_\infty$ and $T(x_\infty)$ are well-defined as $n$-dimensional vectors with components that are either real or are equal to $-\infty$.

Yu [BeY10], [BeY12], [YuB13], which relate to the discounted DP and stochastic shortest path contexts). These algorithms, however, do not use proximal iterations. In any case, an initial vector $x_0$ with $x_0 \geq T(x_0)$ [cf. the assumption of part (b)] may be obtained in some important cases by adding sufficiently large scalars to the components of some given vector $x$. For example, suppose that for some $\mu$, the mapping $T_\mu$ is a sup-norm contraction. Then given any $x \in \Re^n$, it can be shown that the vector $x_0$ with components equal to $x(i) + r$ satisfies $x_0 \geq T_\mu x_0 \geq T(x_0)$, provided that the scalar $r$ is sufficiently large.

The monotone nonincreasing property of the sequence $\{x_k\}$, shown in Prop. 5.2(c), suggests that the algorithm is convergent even when implemented in distributed asynchronous fashion. Indeed this can be shown with an analysis similar to the one given in [BeY10]. Some well-known counterexamples from DP by Williams and Baird [WiB93] suggest that the assumption $x_0 \geq T(x_0)$ is essential for asynchronous convergence (unless the alternative algorithmic framework of [Ber18], Section 2.6.3, is used). Proposition 5.2(c) also shows that even without a guarantee of existence of a fixed point of $T$ within $X$, we have $x_k \downarrow x_\infty$, where $x_\infty$ is some vector that may have some components that are equal to $-\infty$. For an example of this type, consider the one-dimensional problem of finding a fixed point of the mapping

$$T(x) = \min_{\mu \in (0,1]} \big\{(1-\mu^2)x - \mu\big\}.$$

Then Assumption 5.1 is satisfied with $X = \Re$, we have $x_\mu = -1/\mu$, $x^* = -\infty$, $x_k \downarrow x^*$ starting from any $x_0 \in \Re$, while $T$ has no real-valued fixed point.

To see what may happen when Assumption 5.1(c) is not satisfied, let $X = \Re$ and consider the mapping $T$ given by

$$T(x) = \min\{1, x\},$$

which is of the form (5.1) but has multiple fixed points, thus violating the conclusion of Prop. 5.2(a). Here $\mathcal{M}$ consists of two vectors: one is $\hat{\mu}$ with $T_{\hat{\mu}} x = 1$, which is proper, and the other is $\bar{\mu}$ with $T_{\bar{\mu}} x = x$, which is improper but does not satisfy Assumption 5.1(c).

### 5.2 A Linearized Proximal Algorithm Under a Contraction Assumption

We will now discuss a randomized version of the linearized proximal algorithm (5.2)-(5.3), which we will analyze under an alternative set of assumptions. While we remove the monotonicity Assumption 5.1(b) and the restriction $x_0 \geq T(x_0)$, we introduce a finiteness assumption on $\mathcal{M}$, and a contraction assumption on $T_\mu$ and $T$. Thus the components of $T$ are required to be concave polyhedral functions of $x$. We will assume the following.

**Assumption 5.2:** Let $\|\cdot\|$ be some norm in $\Re^n$, and assume the following:

(a) The set $\mathcal{M}$ is finite.

(b) The mappings $T_\mu$, $\mu \in \mathcal{M}$, and $T$ are contractions with respect to the norm $\|\cdot\|$, with modulus $\rho$, and fixed points $x_\mu$ and $x^*$, respectively.

In the special case where the mappings $T_\mu$, $\mu \in \mathcal{M}$, are contractions, with respect to a weighted sup-norm

$$\|x\| = \max_{i=1,\ldots,n} \frac{|x^i|}{v^i}, \qquad x = (x^1, \ldots, x^n) \in \Re^n,$$

where $(v^1, \ldots, v^n)$ is a vector of positive scalar weights, we have that $T$ is also a contraction with respect to the same weighted sup-norm. This is well known in the theory of abstract DP (see [Den67], [BeS78], [Ber18]). For a verification, we write for all $x = (x^1, \ldots, x^n)$, $\tilde{x} = (\tilde{x}^1, \ldots, \tilde{x}^n)$, and $\mu \in \mathcal{M}$,

$$(T_\mu x)^i \leq (T_\mu \tilde{x})^i + \rho \|x - \tilde{x}\| \, v^i, \qquad \forall \; i = 1, \ldots, n.$$

By taking infimum of both sides over $\mu \in \mathcal{M}$, we have

$$\big(T(x)\big)^i - \big(T(\tilde{x})\big)^i \leq \rho \|x - \tilde{x}\| \, v^i, \qquad \forall \; i = 1, \ldots, n.$$

Reversing the roles of $x$ and $\tilde{x}$, we also have

$$\frac{\big(T(\tilde{x})\big)^i - \big(T(x)\big)^i}{v^i} \leq \rho \|x - \tilde{x}\|, \qquad \forall \; i = 1, \ldots, n.$$

By combining the preceding two relations, and taking the maximum of the left side over $i = 1, \ldots, n$, we obtain $\big\|T(x) - T(\tilde{x})\big\| \leq \rho \|x - \tilde{x}\|$. For other norms, however, $T$ may not be a contraction even if all the mappings $T_\mu$ are, so its contraction property must be verified separately.

To construct a convergent linearized proximal algorithm for finding the fixed point of $T$, we address a fundamental difficulty of the algorithm (5.2)-(5.3), which is that the iteration $x_{k+1} = T_{\mu_k}^{(\lambda)} x_k$ [cf. Eq. (5.6)] approaches the fixed point $x_{\mu_k}$ of $T_{\mu_k}$, thus constantly aiming at a "moving target" that depends on $k$. This can cause an oscillatory behavior, whereby the algorithm can get locked into a repeating cycle. We hence introduce modifications that involve a randomization, which allows the algorithm to recover from such a cycle.

In particular, we introduce a fixed probability $p \in (0, 1)$, and for a given $\lambda \in (0, 1)$, we consider the following algorithm

$$x_{k+1} = \begin{cases} T_{\mu_k}^{(\lambda)} x_k & \text{with probability } p, \\ T(x_k) & \text{with probability } 1 - p, \end{cases} \tag{5.14}$$

where $\mu_k$ is updated according to

$$\mu_k \in \arg\min_{\mu \in \mathcal{M}} T_\mu x_k, \tag{5.15}$$

each time the iteration $x_{k+1} = T(x_k)$ is performed. A similarly randomized version of the algorithm (5.2), (5.6) was introduced in Section 2.6.2 of the abstract DP monograph [Ber18]. It has the form

$$x_{k+1} = \begin{cases} T_{\mu_k}^{m_k} x_k & \text{with probability } p, \\ T(x_k) & \text{with } 1 - p, \end{cases} \tag{5.16}$$

where $m_k$ is a positive integer from a bounded range, and $\mu_k$ is updated by Eq. (5.15) each time the iteration $x_{k+1} = T(x_k)$ is performed. A variant of this algorithm is to select to each iteration $k$ a positive integer $m_k$ by randomization using a probability distribution $\big\{p(1), p(2), \ldots\big\}$ over the positive integers, with $p(1) > 0$, and update $\mu_k$ by Eq. (5.15) each time $m_k = 1$. However, the analysis of [Ber18] assumes that the mappings $T_\mu$ are weighted sup-norm contractions, while allowing also for asynchronous operation. By contrast, here the contraction norm need not be a sup-norm, while no asynchrony in updating the components of $x_k$ is allowed.

An important fact (which relies strongly on the finiteness of $\mathcal{M}$) is given in the following proposition.

**Proposition 5.3:** Let Assumption 5.2 hold and let $\mathcal{M}^*$ be the set of all $\mu \in \mathcal{M}$ such that $T_\mu x^* = T(x^*)$. Then for all $\mu \in \mathcal{M}^*$, we have $x_\mu = x^*$. Moreover, there exists an open sphere $S_{x^*}$ that is centered at $x^*$ and is such that for all $x \in S_{x^*}$ we have $T_\mu x = T(x)$ only if $\mu \in \mathcal{M}^*$.

**Proof:** The relation $T_\mu x^* = T(x^*) = x^*$ implies that $x^*$ is the unique fixed point $x_\mu$ of $T_\mu$. The remaining statement follows from the form of the components of $T$ as the minimum of a finite number of linear functions; cf. Eq. (5.1) and Assumption 5.2(a). **Q.E.D.**

The preceding proposition illustrates the key idea of the algorithm (5.14)-(5.15), which is that for $\mu \in \mathcal{M}^*$, the mappings $T_\mu^{(\lambda)}$ are all contractions with a common fixed point equal to $x^*$, the fixed point of $T$, and modulus

$$\frac{\rho(1-\lambda)}{1-\rho\lambda} < \rho$$

[cf. the definition (1.4) of $T_\mu^{(\lambda)}$]. Thus within the sphere $S_{x^*}$, the iterates (5.14) aim consistently at $x^*$. Moreover, because of the randomization, the algorithm is guaranteed (with probability one) to eventually enter the sphere $S_{x^*}$. We have the following proposition.

**Proposition 5.4:** Under Assumption 5.2, for any starting point $x_0$, a sequence $\{x_k\}$ generated by the algorithm (5.14)-(5.15) converges to $x^*$ with probability one.

**Proof:** We will show that $\{x_k\}$ is bounded by showing that for all $k$, we have

$$\max_{\mu\in\mathcal{M}} \|x_k - x_\mu\| \leq \rho^k \max_{\mu\in\mathcal{M}} \|x_0 - x_\mu\| + \frac{2}{1-\rho} \max_{\mu,\mu'\in\mathcal{M}} \|x_\mu - x_{\mu'}\|. \tag{5.17}$$

To this end we will consider separately the two cases where $x_k = T_{\mu_k}^{(\lambda)} x_{k-1}$ (probability $p$) and $x_k = T(x_{k-1})$ (probability $1-p$). In the former case, we have for all $\mu \in \mathcal{M}$,

$$\begin{aligned}
\|x_k - x_\mu\| &\leq \|x_k - x_{\mu^{k-1}}\| + \|x_{\mu^{k-1}} - x_\mu\| \\
&= \|T_{\mu_k}^{(\lambda)} x_{k-1} - x_{\mu^{k-1}}\| + \|x_{\mu^{k-1}} - x_\mu\| \\
&\leq \rho\|x_{k-1} - x_{\mu^{k-1}}\| + \|x_{\mu^{k-1}} - x_\mu\| \\
&\leq \rho\big(\|x_{k-1} - x_\mu\| + \|x_\mu - x_{\mu^{k-1}}\|\big) + \|x_{\mu^{k-1}} - x_\mu\| \\
&\leq \rho \max_{\mu\in\mathcal{M}} \|x_{k-1} - x_\mu\| + 2 \max_{\mu,\mu'\in\mathcal{M}} \|x_\mu - x_{\mu'}\|,
\end{aligned}$$

and finally, for all $k$,

$$\max_{\mu\in\mathcal{M}} \|x_k - x_\mu\| \leq \rho \max_{\mu\in\mathcal{M}} \|x_{k-1} - x_\mu\| + 2 \max_{\mu,\mu'\in\mathcal{M}} \|x_\mu - x_{\mu'}\|. \tag{5.18}$$

A similar calculation shows that this inequality holds also in the complementary case where $x_k = T(x_{k-1})$. By using the relation (5.18) repeatedly, we obtain Eq. (5.17). Thus in conclusion, we have $\{x_k\} \subset B$, where $B$ is the bounded set

$$B = \left\{ x \;\middle|\; \max_{\mu\in\mathcal{M}} \|x - x_\mu\| \leq \max_{\mu\in\mathcal{M}} \|x_0 - x_\mu\| + \frac{2}{1-\rho} \max_{\mu,\mu'\in\mathcal{M}} \|x_\mu - x_{\mu'}\| \right\}.$$

Each time the iteration $x_{k+1} = T(x_k)$ is performed, the distance of the iterate $x_k$ to $x^*$ is reduced by a factor $\rho$, i.e., $\|x_{k+1} - x^*\| \leq \rho\|x_k - x^*\|$. In view of our randomization scheme, the algorithm is guaranteed (with probability one) to eventually execute a sufficient number of contiguous iterations $x_{k+1} = T(x_k)$ to enter the sphere $S_{x^*}$. Once this happens, all the subsequent iterations generate $\mu_k$ within $\mathcal{M}^*$, and aim towards $x^*$ in the sense that they reduce the distance $\|x_k - x^*\|$ by a factor of at least $\rho$. Thus the iterates $x_k$ will eventually enter and remain within $S_{x^*}$, while $x_k \to x^*$. **Q.E.D.**

Note that the proof just given also applies to the algorithm (5.16). The proof suggests that a potentially important issue is the choice of the randomization probability $p$ in the linearized algorithms (5.16) and (5.14). If $p$ is too large, convergence may be slow because oscillatory behavior may go unchecked for a long time. On the other hand if $p$ is small, a correspondingly large number of fixed point iterations $x_{k+1} = T(x_k)$ may be performed, and the hoped for benefits of the use of the proximal iterations may be lost. Adaptive schemes which adjust $p$ based on algorithmic progress may be an interesting possibility for addressing this issue.

Finally, let us note that the analysis of this subsection applies to the algorithms (5.16) and (5.14) for other definitions of $T$, not necessarily involving minimization as in Eq. (5.1). For example, the components $T(i,x)$ of $T(x)$ may be defined by

$$T(i,x) = \max_{\mu(i)\in M(i)} \big\{a\big(i,\mu(i)\big)'x + b\big(i,\mu(i)\big)\big\}, \qquad x \in \Re^n,\ i = 1,\ldots,n, \tag{5.19}$$

or by

$$T(i,x) = \min_{\mu(i)\in M(i)} \max_{\nu(i)\in N(i)} \big\{a\big(i,\mu(i),\nu(i)\big)'x + b\big(i,\mu(i),\nu(i)\big)\big\}, \qquad x \in \Re^n,\ i = 1,\ldots,n, \tag{5.20}$$

where $N(i)$ is a given finite set for each $i$, and $a\big(i,\mu(i),\nu(i)\big)$ and $b\big(i,\mu(i),\nu(i)\big)$ are vectors and scalars, respectively, that depend on $i, \mu(i), \nu(i)$.

In the case of the mapping (5.19), the mapping $T_\mu$ is defined by $T_\mu = A_\mu x + b_\mu$ [cf. Eq. (5.4)], and we have

$$T(x) = \max_{\mu\in\mathcal{M}} T_\mu x.$$

Moreover, $\mu_k$ is updated by maximization, so the algorithm is

$$x_{k+1} = \begin{cases} T_{\mu_k}^{(\lambda)} x_k & \text{with probability } p, \\ T(x_k) & \text{with probability } 1-p, \end{cases} \tag{5.21}$$

where

$$\mu_k \in \arg\max_{\mu\in\mathcal{M}} T_\mu x_k.$$

In the case of the mapping (5.20), the mapping $T_\mu$ has (nonlinear) components defined by

$$T_\mu(i,x) = \max_{\nu(i)\in N(i)} \big\{a\big(i,\mu(i),\nu(i)\big)'x + b\big(i,\mu(i),\nu(i)\big)\big\}, \qquad x \in \Re^n,\ i = 1,\ldots,n,$$

and we have

$$T(x) = \min_{\mu\in\mathcal{M}} T_\mu x.$$

The algorithm is again defined by Eq. (5.21) with $\mu_k$ updated according to

$$\mu_k \in \arg\min_{\mu\in\mathcal{M}} T_\mu x_k.$$

In both cases the convergence result of Prop. 5.2 holds as stated. The mapping (5.20) arises among others in sequential games, which admit a DP treatment (see, e.g., the books [FiV96], [Ber12], or the survey [RaF91]).

Another interesting and related context arises when we want to solve a fixed point equation of the form

$$x = WT(x),$$

where $W$ is an $n \times n$ matrix, and $T$ as earlier has the form

$$T(x) = \min_{\mu \in \mathcal{M}} T_\mu x,$$

where $T_\mu$ is the linear mapping $T_\mu = A_\mu x + b_\mu$ [cf. Eq. (5.4)]. An example is when $W$ is a projection on a subspace spanned by a set of basis functions [cf. Eq. (3.1)]. This context occurs in approximate DP, as discussed in Section 3, where the mappings $T_\mu$ and $T$ are sup-norm contractions, but the composition mapping $WT$ need not be a contraction of any kind, even when $W$ is nonexpansive with respect to some norm. The reason is a potential "norm mismatch" problem, whereby $W$ is nonexpansive with respect to one norm, but $T$ is a contraction with respect to another norm (such examples are well-known; see de Farias and Van Roy [DFV00]). However, in the favorable case where all the mappings $(WT_\mu)^{(\lambda)}$, $\mu \in \mathcal{M}$, as well as the mapping $WT$ are contractions with respect to some norm, the line of analysis of the present section applies to the algorithm

$$x_{k+1} = \begin{cases} (WT_{\mu_k})^{(\lambda)} x_k & \text{with probability } p, \\ WT(x_k) & \text{with probability } 1-p, \end{cases} \tag{5.22}$$

where

$$\mu_k \in \arg\min_{\mu \in \mathcal{M}} T_\mu x_k.$$

In particular, assuming that the set $\mathcal{M}$ is finite, we can simply modify the proof of Prop. 5.3 to show that a sequence $\{x_k\}$ generated by this algorithm converges to the fixed point $x^*$ of $WT$ with probability one. For further discussion of this context, the associated convergence issues, and alternative choices of the matrix $W$, we refer to the author's survey [Ber11b] (Section 3.4) and the textbook [Ber12a].

## 6. CONCLUDING REMARKS

In this paper we have shown that proximal and temporal difference methods for linear fixed point problems are closely related, and their implementations can benefit from each other, in both the exact and the approximate simulation-based setting. In particular, within the context of DP, the TD($\lambda$) algorithm for exact policy evaluation, can be written as the stochastic proximal algorithm

$$x_{k+1} = x_k + \gamma_k \Big( sample\big(P^{(c)} x_k\big) - x_k \Big),$$

for solving the linear Bellman equation $x = Tx$ corresponding to a policy [in view of Eq. (1.5), and taking into account the fact that $(T^{(\lambda)} x_k - x_k)$ is the product of $(P^{(c)} x_k - x_k)$ with the scalar $1/\lambda$, where $\lambda = \frac{c}{c+1}$; cf. Fig. 1.1]. Of course, the probabilistic mechanism used to obtain the sample in the preceding equation is an important algorithmic issue, extensively researched in the approximate DP literature, but our analysis has not dealt with this (or any other issues that relate to stochastic proximal algorithms). Another important issue, which we have not fully explored, is how to connect the TD($\lambda$) algorithm for approximate policy evaluation and the projected proximal algorithm $x_{k+1} = \Pi P^{(c)} x_k$ of Section 3.

Our Assumption 1.1 is satisfied in broad classes of problems, including problems involving a contraction, and policy evaluation in exact and approximate DP. However, even when the assumption is not satisfied and $I - A$ is just invertible, the proximal equation

$$x = P^{(c)} x = \left( \frac{c+1}{c} I - A \right)^{-1} \left( b + \frac{1}{c} x \right), \tag{6.1}$$

makes sense and has the same solution as the original equation $x = Ax + b$, provided the inverse in Eq. (6.1) exists. In this case both the proximal equation $x = P^{(c)}x$ and its projected version $x = \Pi P^{(c)}x$ can be solved by simulation methods, which have been described in the papers [BeY07], [Yu12], [WaB13], [WaB14]; see also the textbook [Ber12a], Section 7.3.

Aside from the conceptual and analytical value of the connection between proximal and temporal difference methods, we have shown that under our assumptions, a tangible improvement of the proximal algorithm is possible at no cost. This improvement is obtained by a simple extrapolation of the proximal iterate, and provides a guaranteed acceleration of convergence (not just guaranteed convergence, like alternative extrapolation schemes). Moreover, this improvement carries over to nonlinear fixed point problems, as we have shown in Section 4. In addition, our methodology extends naturally to forward-backward splitting and proximal gradient algorithms.

To extend the connection between proximal and temporal difference algorithms, we have also introduced some proximal-like algorithms for nonlinear fixed point problems. These algorithms are based on linearization, bear a resemblance with Newton's method, and admit temporal differences-based implementations. They are inspired by distributed asynchronous policy iteration methods for infinite horizon DP, given in the papers [BeY10], [BeY12], [YuB13], and the monograph [Ber18], Section 2.6. Making a stronger connection of these methods and temporal difference methods [including TD($\lambda$)] for solving nonlinear Bellman equations exactly or approximately appears to be a fruitful direction of research.

Some computational experience with the use of simulation to solve large linear systems, beyond those arising in DP, will be helpful in quantifying the potential benefits of the ideas of this paper. Another interesting question is how to generalize the methods of this paper from the linear equation context to the solution of linear variational inequalities, possibly with a large number of constraints, where both the proximal algorithm and multistep DP-type methods have been applied; see the papers [WaB15] and [Ber11a].

## 7. REFERENCES

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

[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, NY.

[BDM14] Boutsidis, C., Drineas, P., and Magdon-Ismail, M., 2014. "Near-Optimal Column-Based Matrix Reconstruction," SIAM J. on Computing, Vol. 43, pp. 687-717.

[BaC11] Bauschke, H. H., and Combettes, P. L., 2011. Convex Analysis and Monotone Operator Theory in Hilbert Spaces, Springer, NY.

[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.

[BeS78] Bertsekas, D. P., and Shreve, S. E., 1978. Stochastic Optimal Control: The Discrete Time Case, Academic Press, N. Y.; may be downloaded from http://web.mit.edu/dimitrib/www/home.html

[BeT91] Bertsekas, D. P., and Tsitsiklis, J. N., 1991. "An Analysis of Stochastic Shortest Path Problems," Math. of OR, Vol. 16, pp. 580-595.

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

[BeT10] Beck, A., and Teboulle, M., 2010. "Gradient-Based Algorithms with Applications to Signal-Recovery Prob-

lems," in Convex Optimization in Signal Processing and Communications (Y. Eldar and D. Palomar, eds.), Cambridge Univ. Press, pp. 42-88.

[BeY07] Bertsekas, D. P., and Yu, H., 2007. "Solution of Large Systems of Equations Using Approximate Dynamic Programming Methods," Lab. for Information and Decision Systems Report LIDS-P-2754, MIT.

[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.

[BeY10] Bertsekas, D. P., and Yu, H., 2010. "Asynchronous Distributed Policy Iteration in Dynamic Programming," Proc. of Allerton Conf. on Communication, Control and Computing, Allerton Park, Ill, pp. 1368-1374.

[BeY12] Bertsekas, D. P., and Yu, H., 2012. "Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," Math. of OR, Vol. 37, pp. 66-94.

[Ber75] Bertsekas, D. P., 1975. "On the Method of Multipliers for Convex Programming," IEEE Transactions on Aut. Control, Vol. 20, pp. 385-388.

[Ber82] Bertsekas, D. P., 1982. Constrained Optimization and Lagrange Multiplier Methods, Academic Press, NY; republished by Athena Scientific, Belmont, MA, 1997; may be downloaded from http://web.mit.edu/dimitrib/www/home.html

[Ber11a] Bertsekas, D. P., 2011. "Temporal Difference Methods for General Projected Equations," IEEE Trans. on Automatic Control, Vol. 56, pp. 2128 - 2139.

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

[Ber12a] Bertsekas, D. P., 2012. Dynamic Programming and Optimal Control: Approximate Dynamic Programming, 4th Edition, Vol. II, Athena Scientific, Belmont, MA.

[Ber12b] Bertsekas, D. P., 2012. "$\lambda$-Policy Iteration: A Review and a New Implementation," in Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, by F. Lewis and D. Liu (eds.), IEEE Press, 2012.

[Ber15] Bertsekas, D. P., 2015. Convex Optimization Algorithms, Athena Scientific, Belmont, MA.

[Ber18] Bertsekas, D. P., 2018. Abstract Dynamic Programming, 2nd Edition, Athena Scientific, Belmont, MA; on-line at http://web.mit.edu/dimitrib/www/home.html.

[Boy02] Boyan, J. A., 2002. "Technical Update: Least-Squares Temporal Difference Learning," Machine Learning, Vol. 49, pp. 1-15.

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

[CeH09] Censor, J., Herman, G. T., and Jiang, M., 2009. "A Note on the Behavior of the Randomized Kaczmarz Algorithm of Strohmer and Vershynin," J. Fourier Analysis and Applications, Vol. 15, pp. 431-436.

[Cur54] Curtiss, J. H., 1954. "A Theoretical Comparison of the Efficiencies of Two Classical Methods and a Monte Carlo Method for Computing One Component of the Solution of a Set of Linear Algebraic Equations," Proc. Symposium on Monte Carlo Methods, pp. 191-233.

[Cur57] Curtiss, J. H., 1957. "A Monte Carlo Methods for the Iteration of Linear Operators," Uspekhi Mat. Nauk, Vol. 12, pp. 149-174.

[DFV00] de Farias, D. P., and Van Roy, B., 2000. "On the Existence of Fixed Points for Approximate Value Iteration and Temporal-Difference Learning," J. of Optimization Theory and Applications, Vol. 105, pp. 589-608.

[DKM06a] Drineas, P., Kannan, R., and Mahoney, M. W., 2006. "Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication," SIAM J. Computing, Vol. 35, pp. 132-157.

[DKM06b] Drineas, P., Kannan, R., and Mahoney, M. W., 2006. "Fast Monte Carlo Algorithms for Matrices II:

Computing a Low-Rank Approximation to a Matrix," SIAM J. Computing, Vol. 36, pp. 158-183.

[DMM06] Drineas, P., Mahoney, M. W., and Muthukrishnan, S., 2006. "Sampling Algorithms for L2 Regression and Applications," Proc. 17th Annual SODA, pp. 1127-1136.

[DMM08] Drineas, P., Mahoney, M. W., and Muthukrishnan, S., 2008. "Relative-Error CUR Matrix Decompositions," SIAM J. Matrix Anal. Appl., Vol. 30, pp. 844-881.

[DMM11] Drineas, P., Mahoney, M. W., Muthukrishnan, S., and Sarlos, T., 2011. "Faster Least Squares Approximation," Numerische Mathematik, Vol. 117, pp. 219-249.

[Den67] Denardo, E. V., 1967. "Contraction Mappings in the Theory Underlying Dynamic Programming," SIAM Review, Vol. 9, pp. 165-177.

[DrL16] Drusvyatskiy, D., and Lewis, A. S., 2016. "Error Bounds, Quadratic Growth, and Linear Convergence of Proximal Methods," arXiv:1602.06661.

[DuR17] Duchi, J. C., and Ryan, F., 2017. "Stochastic Methods for Composite Optimization Problems," arXiv:1703.08570.

[EcB92] Eckstein, J., and Bertsekas, D. P., 1992. "On the Douglas-Rachford Splitting Method and the Proximal Point Algorithm for Maximal Monotone Operators," Math. Programming, Vol. 55, pp. 293-318.

[FaP03] Facchinei, F., and Pang, J.-S., 2003. Finite-Dimensional Variational Inequalities and Complementarity Problems, Springer Verlag, NY.

[FiV96] Filar, J., and Vrieze, K., 1996. Competitive Markov Decision Processes, Springer, N. Y.

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

[FoL50] Forsythe, G. E., and Leibler, R. A., 1950. "Matrix Inversion by a Monte Carlo Method," Mathematical Tables and Other Aids to Computation, Vol. 4, pp. 127-129.

[GMS13] Gabillon, V., Ghavamzadeh, M., and Scherrer, B., 2013. "Approximate Dynamic Programming Finally Performs Well in the Game of Tetris," In Advances in Neural Information Processing Systems, pp. 1754-1762.

[Gab83] Gabay, D., 1983. "Applications of the Method of Multipliers to Variational Inequalities," in M. Fortin and R. Glowinski, eds., Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems, North-Holland, Amsterdam.

[Hal70] Halton, J. H., 1970. "A Retrospective and Prospective Survey of the Monte Carlo Method," SIAM Review, Vol. 12, pp. 1-63.

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

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

[LeL10] Leventhal, D., and Lewis, A. S., 2010. "Randomized Methods for Linear Constraints: Convergence Rates and Conditioning," Mathematics of Operations Research, Vol. 35, pp. 641-654.

[LeL13] Lewis, F. L., and Liu, D., (Eds), 2013. Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, Wiley, Hoboken, N. J.

[LeW08] Lewis, A. S., and Wright, S. J., 2008. "A Proximal Method for Composite Minimization," arXiv:0812.0423.

[LiM79] Lions, P. L., and Mercier, B., 1979. "Splitting Algorithms for the Sum of Two Nonlinear Operators," SIAM J. on Numerical Analysis, Vol. 16, pp. 964-979.

[MKS15] Mnih, V., Kavukcuoglu, K., Silver, D., et al., 2015. "Human-Level Control Through Deep Reinforcement Learning," Nature, Vol. 518, pp. 529-533.

[Mar70] Martinet, B., 1970. "Regularisation d' Inequations Variationnelles par Approximations Successives," Rev. Francaise Inf. Rech. Oper., Vol. 4, pp. 154-158.

[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.

[PaB13] Parikh, N., and Boyd, S., 2013. "Proximal Algorithms," Foundations and Trends in Optimization, Vol. 1, pp. 123-231.

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

[RaF91] Raghavan, T. E. S., and Filar, J. A., 1991. "Algorithms for Stochastic Games – A Survey," ZOR – Methods and Models of Operations Research, Vol. 35, pp. 437-472.

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

[SHM16] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, S., et al. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search," Nature, Vol. 529, pp. 484-489.

[SMG15] Scherrer, B., Ghavamzadeh, M., Gabillon, V., Lesner, B., and Geist, M., 2015. "Approximate Modified Policy Iteration and its Application to the Game of Tetris," J. of Machine Learning Research, Vol. 16, pp. 1629-1676.

[Sam59] Samuel, A. L., 1959. "Some Studies in Machine Learning Using the Game of Checkers," IBM Journal of Research and Development, Vol. 3, 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, Vol. 11, pp. 601-617.

[Sch10] Scherrer, B., 2010. "Should One Compute the Temporal Difference Fixed Point or Minimize the Bellman Residual? The Unified Oblique Projection View," Proc. of 2010 ICML, Haifa, Israel.

[Sch13] Scherrer, B., 2013. "Performance Bounds for $\lambda$-Policy Iteration and Application to the Game of Tetris," J. of Machine Learning Research, Vol. 14, pp. 1181-1227.

[StV09] Strohmer, T., and Vershynin, R., 2009. "A Randomized Kaczmarz Algorithm with Exponential Convergence", J. of Fourier Analysis and Applications, Vol. 15, pp. 262-178.

[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.

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

[TeG96] Tesauro, G., and Galperin, G. R., 1996. "On-Line Policy Improvement Using Monte Carlo Search," presented at the 1996 Neural Information Processing Systems Conference, Denver, CO; also in M. Mozer et al. (eds.), Advances in Neural Information Processing Systems 9, MIT Press (1997).

[Tes94] Tesauro, G. J., 1994. "TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play," Neural Computation, Vol. 6, pp. 215-219.

[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.

[Tse91] Tseng, P., 1991. "Applications of a Splitting Algorithm to Decomposition in Convex Programming and Variational Inequalities," SIAM J. on Control and Optimization, Vol. 29, pp. 119-138.

[VVL13] Vrabie, D., Vamvoudakis, K. G., and Lewis, F. L., 2013. Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles, The Institution of Engineering and Technology, London.

[WaB13] Wang, M., and Bertsekas, D. P., 2013. "Stabilization of Stochastic Iterative Methods for Singular and Nearly Singular Linear Systems," Math. of Operations Research, Vol. 39, pp. 1-30.

[WaB14] Wang, M., and Bertsekas, D. P., 2014. "Convergence of Iterative Simulation-Based Methods for Singular Linear Systems," Stochastic Systems, Vol. 3, pp. 39-96.

[WaB15] Wang, M., and Bertsekas, D. P., 2015. "Incremental Constraint Projection Methods for Variational Inequalities," Math. Programming, Vol. 150, pp. 321-363.

[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.

[YuB06] Yu, H., and Bertsekas, D. P., 2006. "Convergence Results for Some Temporal Difference Methods Based on Least Squares," IEEE Trans. on Aut. Control, Vol. 54, pp. 1515-1531.

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

[YuB12] Yu, H., and Bertsekas, D. P., 2012. "Weighted Bellman Equations and their Applications in Dynamic Programming," Lab. for Information and Decision Systems Report LIDS-P-2876, MIT.

[YuB13] Yu, H., and Bertsekas, D. P., 2013. "Q-Learning and Policy Iteration Algorithms for Stochastic Shortest Path Problems," Annals of Operations Research, Vol. 208, pp. 95-132.

[Yu10] Yu, H., 2010. "Convergence of Least Squares Temporal Difference Methods Under General Conditions," Proc. of the 27th ICML, Haifa, Israel.

[Yu12] Yu, H., 2012. "Least Squares Temporal Difference Methods: An Analysis Under General Conditions," SIAM J. on Control and Optimization, Vol. 50, pp. 3310-3343.