Dynamic Programming and Optimal Control
3rd Edition, Volume II

by

Dimitri P. Bertsekas

Massachusetts Institute of Technology

Chapter 6
Approximate Dynamic Programming

This is an updated version of the research-oriented Chapter 6 on Approximate Dynamic Programming. It will be periodically updated as new research becomes available, and will replace the current Chapter 6 in the book's next printing.

In addition to editorial revisions, rearrangements, and new exercises, the chapter includes an account of new research, which is collected mostly in Sections 6.3 and 6.8. Furthermore, a lot of new material has been added, such as an account of post-decision state simplifications (Section 6.1), regression-based TD methods (Section 6.3), feature scaling (Section 6.3), policy oscillations (Section 6.3), λ-policy iteration and exploration enhanced TD methods, aggregation methods (Section 6.4), new Q-learning algorithms (Section 6.5), and Monte Carlo linear algebra (Section 6.8).

This chapter represents "work in progress." It more than likely contains errors (hopefully not serious ones). Furthermore, its references to the literature are incomplete. Your comments and suggestions to the author at dimitrib@mit.edu are welcome. The date of last revision is given below.

November 11, 2011

6

Approximate Dynamic Programming

Contents

In this chapter we consider approximation methods for challenging, computationally intensive DP problems. We discussed a number of such methods in Chapter 6 of Vol. I and Chapter 1 of the present volume, such as for example rollout and other one-step lookahead approaches. Here our focus will be on algorithms that are mostly patterned after two principal methods of infinite horizon DP: policy and value iteration. These algorithms form the core of a methodology known by various names, such as *approximate dynamic programming*, or *neuro-dynamic programming*, or *reinforcement learning*.

A principal aim of the methods of this chapter is to address problems with very large number of states n. In such problems, ordinary linear algebra operations such as n-dimensional inner products, are prohibitively time-consuming, and indeed it may be impossible to even store an n-vector in a computer memory. Our methods will involve linear algebra operations of dimension much smaller than n, and require only that the components of n-vectors are just generated when needed rather than stored.

Another aim of the methods of this chapter is to address *model-free* situations, i.e., problems where a mathematical model is unavailable or hard to construct. Instead, the system and cost structure may be simulated (think, for example, of a queueing network with complicated but well-defined service disciplines at the queues). The assumption here is that there is a computer program that simulates, for a given control u, the probabilistic transitions from any given state i to a successor state j according to the transition probabilities $p_{ij}(u)$, and also generates a corresponding transition cost $g(i, u, j)$.

Given a simulator, it may be possible to use repeated simulation to calculate (at least approximately) the transition probabilities of the system and the expected stage costs by averaging, and then to apply the methods discussed in earlier chapters. The methods of this chapter, however, are geared towards an alternative possibility, which is much more attractive when one is faced with a large and complex system, and one contemplates approximations. Rather than estimate explicitly the transition probabilities and costs, we will aim to approximate the cost function of a given policy or even the optimal cost-to-go function by generating one or more simulated system trajectories and associated costs, and by using some form of "least squares fit."

Implicit in the rationale of methods based on cost function approximation is of course the hypothesis that a more accurate cost-to-go approximation will yield a better one-step or multistep lookahead policy. This is a reasonable but by no means self-evident conjecture, and may in fact not even be true in a given problem. In another type of method, which we will discuss somewhat briefly, we use simulation in conjunction with a gradient or other method to approximate directly an optimal policy with a policy of a given parametric form. This type of method does not aim at good cost function approximation through which a well-performing policy

may be obtained. Rather it aims directly at finding a policy with good performance.

Let us also mention, two other approximate DP methods, which we have discussed at various points in other parts of the book, but we will not consider further: rollout algorithms (Sections 6.4, 6.5 of Vol. I, and Section 1.3.5 of Vol. II), and approximate linear programming (Section 1.3.4).

Our main focus will be on two types of methods: *policy evaluation algorithms*, which deal with approximation of the cost of a single policy (and can also be embedded within a policy iteration scheme), and *Q-learning algorithms*, which deal with approximation of the optimal cost. Let us summarize each type of method, focusing for concreteness on the finite-state discounted case.

Policy Evaluation Algorithms

With this class of methods, we aim to approximate the cost function $J_\mu(i)$ of a policy μ with a parametric architecture of the form $\tilde{J}(i,r)$, where r is a parameter vector (cf. Section 6.3.5 of Vol. I). This approximation may be carried out repeatedly, for a sequence of policies, in the context of a policy iteration scheme. Alternatively, it may be used to construct an approximate cost-to-go function of a single suboptimal/heuristic policy, which can be used in an on-line rollout scheme, with one-step or multistep lookahead. We focus primarily on two types of methods.†

In the first class of methods, called *direct*, we use simulation to collect samples of costs for various initial states, and fit the architecture $\tilde{J}$ to the samples through some least squares problem. This problem may be solved by several possible algorithms, including linear least squares methods based on simple matrix inversion. Gradient methods have also been used extensively, and will be described in Section 6.2.

The second and currently more popular class of methods is called *indirect*. Here, we obtain r by solving an approximate version of Bellman's equation. We will focus exclusively on the case of a *linear architecture*, where $\tilde{J}$ is of the form Φr, and Φ is a matrix whose columns can be viewed as basis functions (cf. Section 6.3.5 of Vol. I). In an important method of

† In another type of policy evaluation method, often called the *Bellman equation error* approach, which we will discuss briefly in Section 6.8.4, the parameter vector r is determined by minimizing a measure of error in satisfying Bellman's equation; for example, by minimizing over r

$$\|\tilde{J} - T\tilde{J}\|,$$

where $\|\cdot\|$ is some norm. If $\|\cdot\|$ is a Euclidean norm, and $\tilde{J}(i,r)$ is linear in r, this minimization is a linear least squares problem.

this type, we obtain the parameter vector r by solving the equation

$$\Phi r = \Pi T(\Phi r), \tag{6.1}$$

where Π denotes projection with respect to a suitable norm on the subspace of vectors of the form Φr, and T is either the mapping T_μ or a related mapping, which also has J_μ as its unique fixed point [here $\Pi T(\Phi r)$ denotes the projection of the vector $T(\Phi r)$ on the subspace].†

We can view Eq. (6.1) as a form of *projected Bellman equation*. We will show that for a special choice of the norm of the projection, ΠT is a contraction mapping, so the projected Bellman equation has a unique solution Φr^*. We will discuss several iterative methods for finding r^* in Section 6.3. All these methods use simulation and can be shown to converge under reasonable assumptions to r^*, so they produce the same approximate cost function. However, they differ in their speed of convergence and in their suitability for various problem contexts. Here are the methods that we will focus on in Section 6.3 for discounted problems, and also in Sections 6.6-6.8 for other types of problems. They all depend on a parameter $\lambda \in [0, 1]$, whose role will be discussed later.

(1) *TD(λ)* or *temporal differences method*. This algorithm may be viewed as a stochastic iterative method for solving a version of the projected equation (6.1) that depends on λ. The algorithm embodies important ideas and has played an important role in the development of the subject, but in practical terms, it is usually inferior to the next two methods, so it will be discussed in less detail.

(2) *LSTD(λ)* or *least squares temporal differences method*. This algorithm computes and solves a progressively more refined simulation-based approximation to the projected Bellman equation (6.1).

(3) *LSPE(λ)* or *least squares policy evaluation method*. This algorithm is based on the idea of executing value iteration within the lower dimensional space spanned by the basis functions. Conceptually, it has the form

$$\Phi r_{k+1} = \Pi T(\Phi r_k) + \text{ simulation noise}, \tag{6.2}$$

† Another method of this type is based on aggregation (cf. Section 6.3.4 of Vol. I) and is discussed in Section 6.4. This approach can also be viewed as a problem approximation approach (cf. Section 6.3.3 of Vol. I): the original problem is approximated with a related "aggregate" problem, which is then solved exactly to yield a cost-to-go approximation for the original problem. The aggregation counterpart of the equation $\Phi r = \Pi T(\Phi r)$ has the form $\Phi r = \Phi D T(\Phi r)$, where Φ and D are matrices whose rows are restricted to be probability distributions (the aggregation and disaggregation probabilities, respectively).

i.e., the current value iterate $T(\Phi r_k)$ is projected on S and is suitably approximated by simulation. The simulation noise tends to 0 asymptotically, so assuming that ΠT is a contraction, the method converges to the solution of the projected Bellman equation (6.1). There are also a number of variants of LSPE(λ). Both LSPE(λ) and its variants have the same convergence rate as LSTD(λ), because they share a common bottleneck: the slow speed of simulation.

Q-Learning Algorithms

With this class of methods, we aim to compute, without any approximation, the optimal cost function (not just the cost function of a single policy). Q-learning maintains and updates for each state-control pair (i, u) an estimate of the expression that is minimized in the right-hand side of Bellman's equation. This is called the *Q-factor* of the pair (i, u), and is denoted by $Q^*(i, u)$. The Q-factors are updated with what may be viewed as a simulation-based form of value iteration, as will be explained in Section 6.5. An important advantage of using Q-factors is that when they are available, they can be used to obtain an optimal control at any state i simply by minimizing $Q^*(i, u)$ over $u \in U(i)$, so the transition probabilities of the problem are not needed.

On the other hand, for problems with a large number of state-control pairs, Q-learning is often impractical because there may be simply too many Q-factors to update. As a result, the algorithm is primarily suitable for systems with a small number of states (or for aggregated/few-state versions of more complex systems). There are also algorithms that use parametric approximations for the Q-factors (see Section 6.5), although their theoretical basis is generally less solid.

Chapter Organization

Throughout this chapter, we will focus almost exclusively on perfect state information problems, involving a Markov chain with a finite number of states i, transition probabilities $p_{ij}(u)$, and single stage costs $g(i, u, j)$. Extensions of many of the ideas to continuous state spaces are possible, but they are beyond our scope. We will consider first, in Sections 6.1-6.5, the discounted problem using the notation of Section 1.3. Section 6.1 provides a broad overview of cost approximation architectures and their uses in approximate policy iteration. Section 6.2 focuses on direct methods for policy evaluation. Section 6.3 is a long section on a major class of indirect methods for policy evaluation, which are based on the projected Bellman equation. Section 6.4 discusses methods based on aggregation. Section 6.5 discusses Q-learning and its variations, and extends the projected Bellman equation approach to the case of multiple policies, and particularly to optimal stopping problems. Stochastic shortest path and average cost problems

are discussed in Sections 6.6 and 6.7, respectively. Section 6.8 extends and elaborates on the projected Bellman equation approach of Sections 6.3, 6.6, and 6.7, discusses another approach based on the Bellman equation error, and generalizes the aggregation methodology. Section 6.9 describes methods based on parametric approximation of policies rather than cost functions.

6.1 GENERAL ISSUES OF COST APPROXIMATION

Most of the methodology of this chapter deals with approximation of some type of cost function (optimal cost, cost of a policy, Q-factors, etc). The purpose of this section is to highlight the main issues involved, without getting too much into the mathematical details.

We start with general issues of parametric approximation architectures, which we have also discussed in Vol. I (Section 6.3.5). We then consider approximate policy iteration (Section 6.1.2), and the two general approaches for approximate cost evaluation (direct and indirect; Section 6.1.3). In Section 6.1.4, we discuss various special structures that can be exploited to simplify approximate policy iteration. In Sections 6.1.5 and 6.1.6 we provide orientation into the main mathematical issues underlying the methodology, and focus on two of its main components: contraction mappings and simulation.

6.1.1 Approximation Architectures

The major use of cost approximation is for obtaining a one-step lookahead suboptimal policy (cf. Section 6.3 of Vol. I).† In particular, suppose that we use $\tilde{J}(j,r)$ as an approximation to the optimal cost of the finite-state discounted problem of Section 1.3. Here $\tilde{J}$ is a function of some chosen form (the approximation architecture) and r is a parameter/weight vector. Once r is determined, it yields a suboptimal control at any state i via the one-step lookahead minimization

$$\tilde{\mu}(i) = \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}(j,r)\big). \tag{6.3}$$

The degree of suboptimality of $\tilde{\mu}$, as measured by $\|J_{\tilde{\mu}} - J^*\|_\infty$, is bounded by a constant multiple of the approximation error according to

$$\|J_{\tilde{\mu}} - J^*\|_\infty \leq \frac{2\alpha}{1-\alpha}\|\tilde{J} - J^*\|_\infty,$$

† We may also use a multiple-step lookahead minimization, with a cost-to-go approximation at the end of the multiple-step horizon. Conceptually, single-step and multiple-step lookahead approaches are similar, and the cost-to-go approximation algorithms of this chapter apply to both.

as shown in Prop. 1.3.7. This bound is qualitative in nature, as it tends to be quite conservative in practice.

An alternative possibility is to obtain a parametric approximation $\tilde{Q}(i,u,r)$ of the Q-factor of the pair (i,u), defined in terms of the optimal cost function J^* as

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big).$$

Since $Q^*(i,u)$ is the expression minimized in Bellman's equation, given the approximation $\tilde{Q}(i,u,r)$, we can generate a suboptimal control at any state i via

$$\tilde{\mu}(i) = \arg\min_{u \in U(i)} \tilde{Q}(i,u,r).$$

The advantage of using Q-factors is that in contrast with the minimization (6.3), the transition probabilities $p_{ij}(u)$ are not needed in the above minimization. Thus Q-factors are better suited to the model-free context.

Note that we may similarly use approximations to the cost functions J_μ and Q-factors $Q_\mu(i,u)$ of specific policies μ. A major use of such approximations is in the context of an approximate policy iteration scheme; see Section 6.1.2.

The choice of architecture is very significant for the success of the approximation approach. One possibility is to use the *linear* form

$$\tilde{J}(i,r) = \sum_{k=1}^{s} r_k \phi_k(i), \tag{6.4}$$

where $r = (r_1, \ldots, r_s)$ is the parameter vector, and $\phi_k(i)$ are some known scalars that depend on the state i. Thus, for each state i, the approximate cost $\tilde{J}(i,r)$ is the inner product $\phi(i)'r$ of r and

$$\phi(i) = \begin{pmatrix} \phi_1(i) \\ \vdots \\ \phi_s(i) \end{pmatrix}.$$

We refer to $\phi(i)$ as the *feature vector* of i, and to its components as *features* (see Fig. 6.1.1). Thus the cost function is approximated by a vector in the subspace

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

where

$$\Phi = \begin{pmatrix} \phi_1(1) & \ldots & \phi_s(1) \\ \vdots & \vdots & \vdots \\ \phi_1(n) & \ldots & \phi_s(n) \end{pmatrix} = \begin{pmatrix} \phi(1)' \\ \vdots \\ \phi(n)' \end{pmatrix}.$$

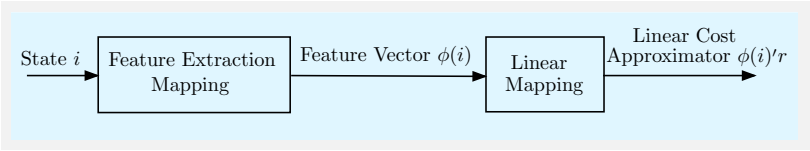

Figure 6.1.1 A linear feature-based architecture. It combines a mapping that extracts the feature vector $\phi(i) = \big(\phi_1(i), \ldots, \phi_s(i)\big)'$ associated with state i, and a parameter vector r to form a linear cost approximator.

We can view the s columns of Φ as basis functions, and Φr as a linear combination of basis functions.

Features, when well-crafted, can capture the dominant nonlinearities of the cost function, and their linear combination may work very well as an approximation architecture. For example, in computer chess (Section 6.3.5 of Vol. I) where the state is the current board position, appropriate features are material balance, piece mobility, king safety, and other positional factors.

Example 6.1.1 (Polynomial Approximation)

An important example of linear cost approximation is based on polynomial basis functions. Suppose that the state consists of q integer components $x_1, \ldots, x_q$, each taking values within some limited range of integers. For example, in a queueing system, x_k may represent the number of customers in the kth queue, where $k = 1, \ldots, q$. Suppose that we want to use an approximating function that is quadratic in the components x_k. Then we can define a total of $1 + q + q^2$ basis functions that depend on the state $x = (x_1, \ldots, x_q)$ via

$$\phi_0(x) = 1, \qquad \phi_k(x) = x_k, \qquad \phi_{km}(x) = x_k x_m, \quad k, m = 1, \ldots, q.$$

A linear approximation architecture that uses these functions is given by

$$\tilde{J}(x, r) = r_0 + \sum_{k=1}^{q} r_k x_k + \sum_{k=1}^{q} \sum_{m=k}^{q} r_{km} x_k x_m,$$

where the parameter vector r has components r_0, r_k, and r_{km}, with $k = 1, \ldots, q$, $m = k, \ldots, q$. In fact, any kind of approximating function that is polynomial in the components $x_1, \ldots, x_q$ can be constructed similarly.

It is also possible to combine feature extraction with polynomial approximations. For example, the feature vector $\phi(i) = \big(\phi_1(i), \ldots, \phi_s(i)\big)'$ transformed by a quadratic polynomial mapping, leads to approximating functions of the form

$$\tilde{J}(i, r) = r_0 + \sum_{k=1}^{s} r_k \phi_k(i) + \sum_{k=1}^{s} \sum_{\ell=1}^{s} r_{k\ell} \phi_k(i) \phi_\ell(i),$$

where the parameter vector r has components r_0, r_k, and $r_{k\ell}$, with $k, \ell = 1, \ldots, s$. This function can be viewed as a linear cost approximation that uses the basis functions

$$w_0(i) = 1, \qquad w_k(i) = \phi_k(i), \qquad w_{k\ell}(i) = \phi_k(i)\phi_\ell(i), \quad k, \ell = 1, \ldots, s.$$

Example 6.1.2 (Interpolation)

A common type of approximation of a function J is based on interpolation. Here, a set I of special states is selected, and the parameter vector r has one component r_i per state $i \in I$, which is the value of J at i:

$$r_i = J(i), \qquad i \in I.$$

The value of J at states $i \notin I$ is approximated by some form of interpolation using r.

Interpolation may be based on geometric proximity. For a simple example that conveys the basic idea, let the system states be the integers within some interval, let I be a subset of special states, and for each state i let $\underline{i}$ and $\bar{i}$ be the states in I that are closest to i from below and from above. Then for any state i, $\tilde{J}(i, r)$ is obtained by linear interpolation of the costs $r_{\underline{i}} = J(\underline{i})$ and $r_{\bar{i}} = J(\bar{i})$:

$$\tilde{J}(i, r) = \frac{i - \underline{i}}{\bar{i} - \underline{i}}\, r_{\underline{i}} + \frac{\bar{i} - i}{\bar{i} - \underline{i}}\, r_{\bar{i}}.$$

The scalars multiplying the components of r may be viewed as features, so the feature vector of i above consists of two nonzero features (the ones corresponding to $\underline{i}$ and $\bar{i}$), with all other features being 0. Similar examples can be constructed for the case where the state space is a subset of a multidimensional space (see Example 6.3.13 of Vol. I).

A generalization of the preceding example is approximation based on aggregation; see Section 6.3.4 of Vol. I and the subsequent Section 6.4 in this chapter. There are also interesting nonlinear approximation architectures, including those defined by neural networks, perhaps in combination with feature extraction mappings (see Bertsekas and Tsitsiklis [BeT96], or Sutton and Barto [SuB98] for further discussion). In this chapter, we will mostly focus on the case of linear architectures, because many of the policy evaluation algorithms of this chapter are valid only for that case.

We note that there has been considerable research on automatic basis function generation approaches (see e.g., Keller, Mannor, and Precup [KMP06], and Jung and Polani [JuP07]). Moreover it is possible to use standard basis functions which may be computed by simulation (perhaps with simulation error). The following example discusses this possibility.

Example 6.1.3 (Krylov Subspace Generating Functions)

We have assumed so far that the columns of Φ, the basis functions, are known, and the rows $\phi(i)'$ of Φ are explicitly available to use in the various simulation-based formulas. We will now discuss a class of basis functions that may not be available, but may be approximated by simulation in the course of various algorithms. For concreteness, let us consider the evaluation of the cost vector

$$J_\mu = (I - \alpha P_\mu)^{-1} g_\mu$$

of a policy μ in a discounted MDP. Then J_μ has an expansion of the form

$$J_\mu = \sum_{t=0}^{\infty} \alpha^t P_\mu^t g_\mu.$$

Thus $g_\mu, P_\mu g_\mu, \ldots, P_\mu^s g_\mu$ yield an approximation based on the first $s+1$ terms of the expansion, and seem suitable choices as basis functions. Also a more general expansion is

$$J_\mu = J + \sum_{t=0}^{\infty} \alpha^t P_\mu^t q,$$

where J is any vector in $\Re^n$ and q is the residual vector

$$q = T_\mu J - J = g_\mu + \alpha P_\mu J - J;$$

this can be seen from the equation $J_\mu - J = \alpha P_\mu (J_\mu - J) + q$. Thus the basis functions $J, q, P_\mu q, \ldots, P_\mu^{s-1} q$ yield an approximation based on the first $s+1$ terms of the preceding expansion.

Generally, to implement various methods in subsequent sections with basis functions of the form $P_\mu^m g_\mu$, $m \geq 0$, one would need to generate the ith components $(P_\mu^m g_\mu)(i)$ for any given state i, but these may be hard to calculate. However, it turns out that one can use instead single sample approximations of $(P_\mu^m g_\mu)(i)$, and rely on the averaging mechanism of simulation to improve the approximation process. The details of this are beyond our scope and we refer to the original sources (Bertsekas and Yu [BeY07], [BeY09]) for further discussion and specific implementations.

We finally mention the possibility of optimal selection of basis functions within some restricted class. In particular, consider an approximation subspace

$$S_\theta = \big\{ \Phi(\theta) r \mid r \in \Re^s \big\},$$

where the s columns of the $n \times s$ matrix Φ are basis functions parametrized by a vector θ. Assume that for a given θ, there is a corresponding vector $r(\theta)$, obtained using some algorithm, so that $\Phi(\theta) r(\theta)$ is an approximation of a cost function J (various such algorithms will be presented later in this chapter). Then we may wish to select θ so that some measure of approximation quality is optimized. For example, suppose that we can

compute the true cost values $J(i)$ (or more generally, approximations to these values) for a subset of selected states I. Then we may determine θ so that

$$\sum_{i \in I} \big(J(i) - \phi(i,\theta)'r(\theta)\big)^2$$

is minimized, where $\phi(i,\theta)'$ is the ith row of $\Phi(\theta)$. Alternatively, we may determine θ so that the norm of the error in satisfying Bellman's equation,

$$\big\|\Phi(\theta)r(\theta) - T\big(\Phi(\theta)r(\theta)\big)\big\|^2,$$

is minimized. Gradient and random search algorithms for carrying out such minimizations have been proposed in the literature (see Menache, Mannor, and Shimkin [MMS06], and Yu and Bertsekas [YuB09]).

6.1.2 Approximate Policy Iteration

Let us consider a form of approximate policy iteration, where we compute simulation-based approximations $\tilde{J}(\cdot, r)$ to the cost functions J_μ of stationary policies μ, and we use them to compute new policies based on (approximate) policy improvement. We impose no constraints on the approximation architecture, so $\tilde{J}(i,r)$ may be linear or nonlinear in r.

Suppose that the current policy is μ, and for a given r, $\tilde{J}(i,r)$ is an approximation of $J_\mu(i)$. We generate an "improved" policy $\overline{\mu}$ using the formula

$$\overline{\mu}(i) = \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}(j,r)\big), \qquad \text{for all } i. \tag{6.5}$$

The method is illustrated in Fig. 6.1.2. Its theoretical basis was discussed in Section 1.3 (cf. Prop. 1.3.6), where it was shown that if the policy evaluation is accurate to within δ (in the sup-norm sense), then for an α-discounted problem, the method will yield in the limit (after infinitely many policy evaluations) a stationary policy that is optimal to within

$$\frac{2\alpha\delta}{(1-\alpha)^2},$$

where α is the discount factor. Experimental evidence indicates that this bound is usually conservative. Furthermore, often just a few policy evaluations are needed before the bound is attained.

When the sequence of policies obtained actually converges to some $\hat{\mu}$, then it can be proved that $\hat{\mu}$ is optimal to within

$$\frac{2\alpha\delta}{1-\alpha}$$

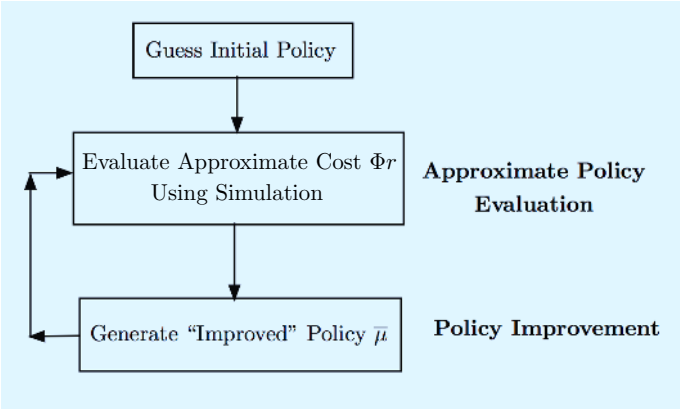

Figure 6.1.2 Block diagram of approximate policy iteration.

(see Section 6.3.8 and also Section 6.4.2, where it is shown that if policy evaluation is done using an aggregation approach, the generated sequence of policies does converge).

A simulation-based implementation of the algorithm is illustrated in Fig. 6.1.3. It consists of four parts:

(a) The *simulator*, which given a state-control pair (i, u), generates the next state j according to the system's transition probabilities.

(b) The *decision generator*, which generates the control $\overline{\mu}(i)$ of the improved policy at the current state i for use in the simulator.

(c) The *cost-to-go approximator*, which is the function $\tilde{J}(j, r)$ that is used by the decision generator.

(d) The *cost approximation algorithm*, which accepts as input the output of the simulator and obtains the approximation $\tilde{J}(\cdot, \overline{r})$ of the cost of $\overline{\mu}$.

Note that there are two policies μ and $\overline{\mu}$, and parameter vectors r and $\overline{r}$, which are simultaneously involved in this algorithm. In particular, r corresponds to the current policy μ, and the approximation $\tilde{J}(\cdot, r)$ is used in the policy improvement Eq. (6.5) to generate the new policy $\overline{\mu}$. At the same time, $\overline{\mu}$ drives the simulation that generates samples to be used by the algorithm that determines the parameter $\overline{r}$ corresponding to $\overline{\mu}$, which will be used in the next policy iteration.

The Issue of Exploration

Let us note an important generic difficulty with simulation-based policy iteration: to evaluate a policy μ, we need to generate cost samples using that policy, but this biases the simulation by underrepresenting states that

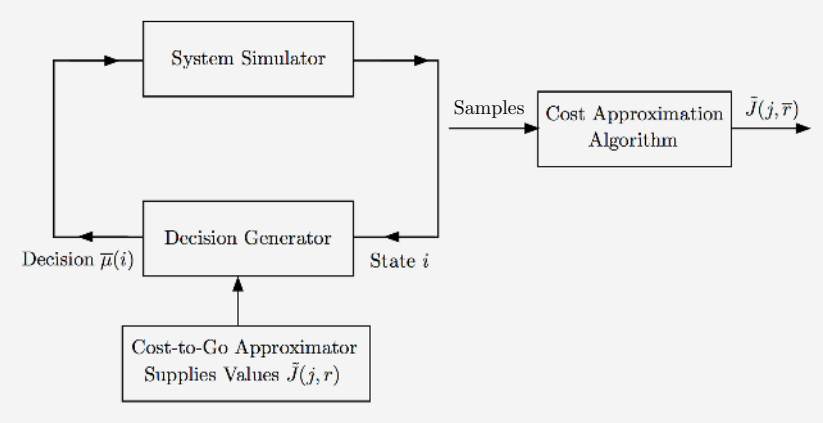

Figure 6.1.3 Simulation-based implementation approximate policy iteration algorithm. Given the approximation $\tilde{J}(i,r)$, we generate cost samples of the "improved" policy $\overline{\mu}$ by simulation (the "decision generator" module). We use these samples to generate the approximator $\tilde{J}(i,\overline{r})$ of $\overline{\mu}$.

are unlikely to occur under μ. As a result, the cost-to-go estimates of these underrepresented states may be highly inaccurate, causing potentially serious errors in the calculation of the improved control policy $\overline{\mu}$ via the policy improvement Eq. (6.5).

The difficulty just described is known as *inadequate exploration* of the system's dynamics because of the use of a fixed policy. It is a particularly acute difficulty when the system is deterministic, or when the randomness embodied in the transition probabilities is "relatively small." One possibility for guaranteeing adequate exploration of the state space is to frequently restart the simulation and to ensure that the initial states employed form a rich and representative subset. A related approach, called *iterative resampling*, is to enrich the sampled set of states in evaluating the current policy μ as follows: derive an initial cost evaluation of μ, simulate the next policy $\overline{\mu}$ obtained on the basis of this initial evaluation to obtain a set of representative states $\overline{S}$ visited by $\overline{\mu}$, and repeat the evaluation of μ using additional trajectories initiated from $\overline{S}$.

Still another frequently used approach is to artificially introduce some extra randomization in the simulation, by occasionally using a randomly generated transition rather than the one dictated by the policy μ (although this may not necessarily work because all admissible controls at a given state may produce "similar" successor states). This and other possibilities to improve exploration will be discussed further in Section 6.3.7.

Limited Sampling/Optimistic Policy Iteration

In the approximate policy iteration approach discussed so far, the policy evaluation of the cost of the improved policy $\overline{\mu}$ must be fully carried out. An alternative, known as *optimistic policy iteration*, is to replace the policy μ with the policy $\overline{\mu}$ after only a few simulation samples have been processed, at the risk of $\tilde{J}(\cdot, \overline{r})$ being an inaccurate approximation of $J_{\overline{\mu}}$.

Optimistic policy iteration has been successfully used, among others, in an impressive backgammon application (Tesauro [Tes92]). However, the associated theoretical convergence properties are not fully understood. As will be illustrated by the discussion of Section 6.3.8 (see also Section 6.4.2 of [BeT96]), optimistic policy iteration can exhibit fascinating and counterintuitive behavior, including a natural tendency for a phenomenon called *chattering*, whereby the generated parameter sequence $\{r_k\}$ converges, while the generated policy sequence oscillates because the limit of $\{r_k\}$ corresponds to multiple policies.

We note that optimistic policy iteration tends to deal better with the problem of exploration discussed earlier, because with rapid changes of policy, there is less tendency to bias the simulation towards particular states that are favored by any single policy.

Approximate Policy Iteration Based on Q-Factors

The approximate policy iteration method discussed so far relies on the calculation of the approximation $\tilde{J}(\cdot, r)$ to the cost function J_μ of the current policy, which is then used for policy improvement using the minimization

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}(j,r)\big).$$

Carrying out this minimization requires knowledge of the transition probabilities $p_{ij}(u)$ and calculation of the associated expected values for all controls $u \in U(i)$ (otherwise a time-consuming simulation of these expected values is needed). A model-free alternative is to compute *approximate Q-factors*

$$\tilde{Q}(i,u,r) \approx \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J_\mu(j)\big), \tag{6.6}$$

and use the minimization

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \tilde{Q}(i,u,r) \tag{6.7}$$

for policy improvement. Here, r is an adjustable parameter vector and $\tilde{Q}(i,u,r)$ is a parametric architecture, possibly of the linear form

$$\tilde{Q}(i,u,r) = \sum_{k=1}^{s} r_k \phi_k(i,u),$$

where $\phi_k(i,u)$ are basis functions that depend on both state and control [cf. Eq. (6.4)].

The important point here is that given the current policy μ, we can construct Q-factor approximations $\tilde{Q}(i,u,r)$ using any method for constructing cost approximations $\tilde{J}(i,r)$. The way to do this is to apply the latter method to the Markov chain whose states are the pairs (i,u), and the probability of transition from (i,u) to (j,v) is

$$p_{ij}(u) \qquad \text{if } v = \mu(j),$$

and is 0 otherwise. This is the probabilistic mechanism by which state-control pairs evolve under the stationary policy μ.

A major concern with this approach is that the state-control pairs (i,u) with $u \neq \mu(i)$ are never generated in this Markov chain, so they are not represented in the cost samples used to construct the approximation $\tilde{Q}(i,u,r)$ (see Fig. 6.1.4). This creates an acute difficulty due to diminished exploration, which must be carefully addressed in any simulation-based implementation. We will return to the use of Q-factors in Section 6.5, where we will discuss exact and approximate implementations of the Q-learning algorithm.

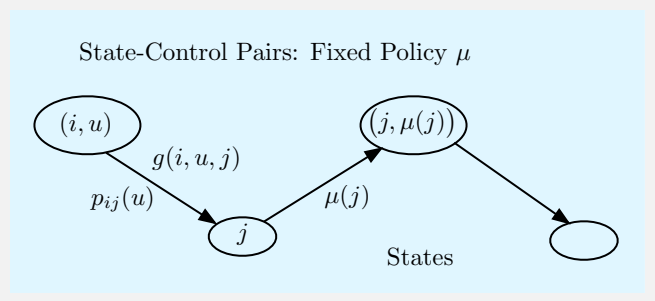

Figure 6.1.4 Markov chain underlying Q-factor-based policy evaluation, associated with policy μ. The states are the pairs (i,u), and the probability of transition from (i,u) to (j,v) is $p_{ij}(u)$ if $v = \mu(j)$, and is 0 otherwise. Thus, after the first transition, the generated pairs are exclusively of the form $(i,\mu(i))$; pairs of the form (i,u), $u \neq \mu(i)$, are not explored.

The Issue of Policy Oscillations

Contrary to exact policy iteration, which converges to an optimal policy in a fairly regular manner, approximate policy iteration may oscillate. By this we mean that after a few iterations, policies tend to repeat in cycles. The associated parameter vectors r may also tend to oscillate. This phenomenon is explained in Section 6.3.8 and can be particularly damaging,

because there is no guarantee that the policies involved in the oscillation are "good" policies, and there is often no way to verify how well they perform relative to the optimal.

We note that oscillations can be avoided and approximate policy iteration can be shown to converge under special conditions that arise in particular when aggregation is used for policy evaluation. These conditions involve certain monotonicity assumptions regarding the choice of the matrix Φ, which are fulfilled in the case of aggregation (see Section 6.3.8, and also Section 6.4.2). However, when Φ is chosen in an unrestricted manner, as often happens in practical applications of the projected equation methods of Section 6.3, policy oscillations tend to occur generically, and often for very simple problems (see Section 6.3.8 for an example).

6.1.3 Direct and Indirect Approximation

We will now preview two general algorithmic approaches for approximating the cost function of a fixed stationary policy μ within a subspace of the form $S = \{\Phi r \mid r \in \Re^s\}$. (A third approach, based on aggregation, uses a special type of matrix Φ and is discussed in Section 6.4.) The first and most straightforward approach, referred to as *direct*, is to find an approximation $\tilde{J} \in S$ that matches best J_μ in some normed error sense, i.e.,

$$\min_{\tilde{J} \in S} \|J_\mu - \tilde{J}\|,$$

or equivalently,

$$\min_{r \in \Re^s} \|J_\mu - \Phi r\|$$

(see the left-hand side of Fig. 6.1.5).† Here, $\|\cdot\|$ is usually some (possibly weighted) Euclidean norm, in which case the approximation problem is a linear least squares problem, whose solution, denoted r^*, can in principle be obtained in closed form by solving the associated quadratic minimization problem. If the matrix Φ has linearly independent columns, the solution is unique and can also be represented as

$$\Phi r^* = \Pi J_\mu,$$

where Π denotes projection with respect to $\|\cdot\|$ on the subspace S.† A major difficulty is that specific cost function values $J_\mu(i)$ can only be estimated

† Note that direct approximation may be used in other approximate DP contexts, such as finite horizon problems, where we use sequential single-stage approximation of the cost-to-go functions J_k, going backwards (i.e., starting with J_N, we obtain a least squares approximation of J_{N-1}, which is used in turn to obtain a least squares approximation of J_{N-2}, etc). This approach is sometimes called *fitted value iteration.*

† In what follows in this chapter, we will not distinguish between the linear operation of projection and the corresponding matrix representation, denoting them both by Π. The meaning should be clear from the context.

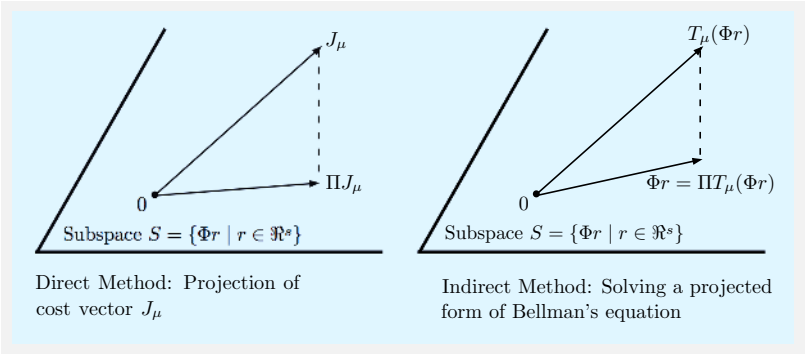

Figure 6.1.5 Two methods for approximating the cost function J_μ as a linear combination of basis functions (subspace S). In the direct method (figure on the left), J_μ is projected on S. In the indirect method (figure on the right), the approximation is found by solving $\Phi r = \Pi T_\mu(\Phi r)$, a projected form of Bellman's equation.

through their simulation-generated cost samples, as we discuss in Section 6.2.

An alternative and more popular approach, referred to as *indirect*, is to approximate the solution of Bellman's equation $J = T_\mu J$ on the subspace S (see the right-hand side of Fig. 6.1.5). An important example of this approach, which we will discuss in detail in Section 6.3, leads to the problem of finding a vector r^* such that

$$\Phi r^* = \Pi T_\mu(\Phi r^*). \tag{6.8}$$

We can view this equation as a *projected form of Bellman's equation*. We will consider another type of indirect approach based on aggregation in Section 6.4.

We note that solving projected equations as approximations to more complex/higher-dimensional equations has a long history in scientific computation in the context of Galerkin methods (see e.g., [Kra72]). For example, some of the most popular finite-element methods for partial differential equations are of this type. However, the use of the Monte Carlo simulation ideas that are central in approximate DP is an important characteristic that differentiates the methods of the present chapter from the Galerkin methodology.

An important fact here is that ΠT_μ *is a contraction, provided we use a special weighted Euclidean norm for projection*, as will be proved in Section 6.3 for discounted problems (Prop. 6.3.1). In this case, Eq. (6.8) has a unique solution, and allows the use of algorithms such as LSPE(λ) and TD(λ), which are discussed in Section 6.3. Unfortunately, the contraction property of ΠT_μ does not extend to the case where T_μ is replaced by

T, the DP mapping corresponding to multiple/all policies, although there are some interesting exceptions, one of which relates to optimal stopping problems and is discussed in Section 6.5.3.

6.1.4 Simplifications

We now consider various situations where the special structure of the problem may be exploited to simplify policy iteration or other approximate DP algorithms.

Problems with Uncontrollable State Components

In many problems of interest the state is a composite (i, y) of two components i and y, and the evolution of the main component i can be directly affected by the control u, but the evolution of the other component y cannot. Then as discussed in Section 1.4 of Vol. I, the value and the policy iteration algorithms can be carried out over a smaller state space, the space of the controllable component i. In particular, we assume that given the state (i, y) and the control u, the next state (j, z) is determined as follows: j is generated according to transition probabilities $p_{ij}(u, y)$, and z is generated according to conditional probabilities $p(z \mid j)$ that depend on the main component j of the new state (see Fig. 6.1.6). Let us assume for notational convenience that the cost of a transition from state (i, y) is of the form $g(i, y, u, j)$ and does not depend on the uncontrollable component z of the next state (j, z). If g depends on z it can be replaced by

$$\hat{g}(i, y, u, j) = \sum_z p(z \mid j) g(i, y, u, j, z)$$

in what follows.

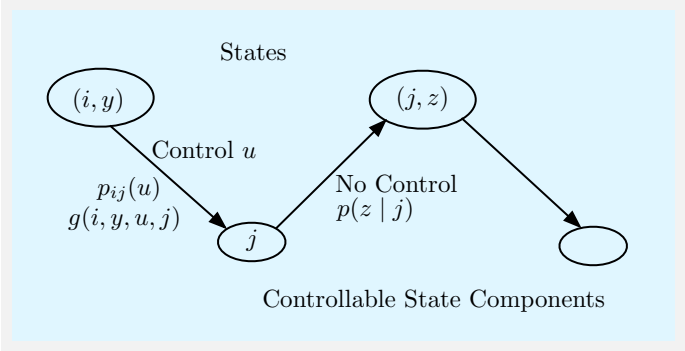

Figure 6.1.6 States and transition probabilities for a problem with uncontrollable state components.

For an α-discounted problem, consider the mapping $\hat{T}$ defined by

$$\begin{aligned}(\hat{T}\hat{J})(i) &= \sum_y p(y \mid i)(T\hat{J})(i,y) \\ &= \sum_y p(y \mid i) \min_{u \in U(i,y)} \sum_{j=0}^{n} p_{ij}(u,y)\big(g(i,y,u,j) + \alpha \hat{J}(j)\big),\end{aligned}$$

and the corresponding mapping for a stationary policy μ,

$$\begin{aligned}(\hat{T}_\mu \hat{J})(i) &= \sum_y p(y \mid i)(T_\mu J)(i,y) \\ &= \sum_y p(y \mid i) \sum_{j=0}^{n} p_{ij}\big(\mu(i,y),y\big)\big(g\big(i,y,\mu(i,y),j\big) + \alpha \hat{J}(j)\big).\end{aligned}$$

Bellman's equation, defined over the controllable state component i, takes the form

$$\hat{J}(i) = (\hat{T}\hat{J})(i), \qquad \text{for all } i. \tag{6.9}$$

The typical iteration of the simplified policy iteration algorithm consists of two steps:

(a) The policy evaluation step, which given the current policy $\mu^k(i,y)$, computes the unique $\hat{J}_{\mu^k}(i)$, $i = 1, \ldots, n$, that solve the linear system of equations $\hat{J}_{\mu^k} = \hat{T}_{\mu^k}\hat{J}_{\mu^k}$ or equivalently

$$\hat{J}_{\mu^k}(i) = \sum_y p(y \mid i) \sum_{j=0}^{n} p_{ij}\big(\mu^k(i,y)\big)\Big(g\big(i,y,\mu^k(i,y),j\big) + \alpha \hat{J}_{\mu^k}(j)\Big)$$

for all $i = 1, \ldots, n$.

(b) The policy improvement step, which computes the improved policy $\mu^{k+1}(i,y)$, from the equation $\hat{T}_{\mu^{k+1}}\hat{J}_{\mu^k} = \hat{T}\hat{J}_{\mu^k}$ or equivalently

$$\mu^{k+1}(i,y) = \arg \min_{u \in U(i,y)} \sum_{j=0}^{n} p_{ij}(u,y)\big(g(i,y,u,j) + \alpha \hat{J}_{\mu^k}(j)\big),$$

for all (i,y).

Approximate policy iteration algorithms can be similarly carried out in reduced form.

Problems with Post-Decision States

In some stochastic problems, the transition probabilities and stage costs have the special form

$$p_{ij}(u) = q\big(j \mid f(i,u)\big), \tag{6.10}$$

where f is some function and $q\big(\cdot \mid f(i,u)\big)$ is a given probability distribution for each value of $f(i,u)$. In words, the dependence of the transitions on (i,u) comes through the function $f(i,u)$. We may exploit this structure by viewing $f(i,u)$ as a form of state: a *post-decision state* that determines the probabilistic evolution to the next state. An example where the conditions (6.10) are satisfied are inventory control problems of the type considered in Section 4.2 of Vol. I. There the post-decision state at time k is $x_k + u_k$, i.e., the post-purchase inventory, before any demand at time k has been filled.

Post-decision states can be exploited when the stage cost has no dependence on j,† i.e., when we have (with some notation abuse)

$$g(i,u,j) = g(i,u).$$

Then the optimal cost-to-go within an α-discounted context at state i is given by

$$J^*(i) = \min_{u \in U(i)} \Big[g(i,u) + \alpha V^*\big(f(i,u)\big) \Big],$$

while the optimal cost-to-go at post-decision state m (optimal sum of costs of future stages) is given by

$$V^*(m) = \sum_{j=1}^{n} q(j \mid m) J^*(j).$$

In effect, we consider a modified problem where the state space is enlarged to include post-decision states, with transitions between ordinary states and post-decision states specified by f and $q\big(\cdot \mid f(i,u)\big)$ (see Fig. 6.1.7). The preceding two equations represent Bellman's equation for this modified problem.

Combining these equations, we have

$$V^*(m) = \sum_{j=1}^{n} q(j \mid m) \min_{u \in U(j)} \Big[g(j,u) + \alpha V^*\big(f(j,u)\big) \Big], \qquad \forall\ m, \tag{6.11}$$

which can be viewed as Bellman's equation over the space of post-decision states m. This equation is similar to Q-factor equations, but is defined

† If there is dependence on j, one may consider computing, possibly by simulation, (an approximation to) $\overline{g}(i,u) = \sum_{j=1}^{n} p_{ij}(u) g(i,u,j)$, and using it in place of $g(i,u,j)$.

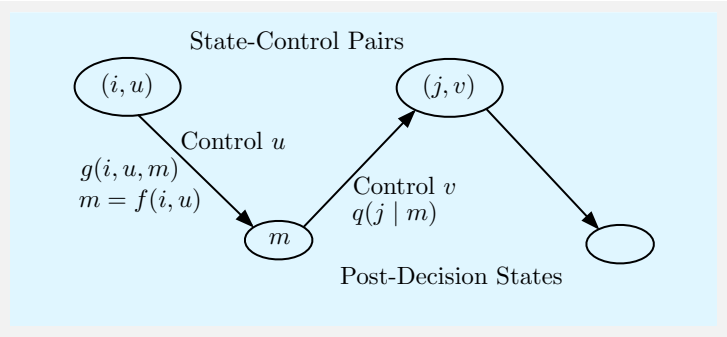

Figure 6.1.7 Modified problem where the post-decision states are viewed as additional states.

over the space of post-decision states rather than the larger space of state-control pairs. The advantage of this equation is that once the function V^* is calculated (or approximated), the optimal policy can be computed as

$$\mu^*(i) = \arg \min_{u \in U(i)} \Big[g(i,u) + \alpha V^*\big(f(i,u)\big) \Big],$$

which does not require the knowledge of transition probabilities and computation of an expected value. It involves a deterministic optimization, and it can be used in a model-free context (as long as the functions g and f are known). This is important if the calculation of the optimal policy is done on-line.

It is straightforward to construct a policy iteration algorithm that is defined over the space of post-decision states. The cost-to-go function V_μ of a stationary policy μ is the unique solution of the corresponding Bellman equation

$$V_\mu(m) = \sum_{j=1}^{n} q(j \mid m) \Big(g\big(j, \mu(j)\big) + \alpha V_\mu\big(f\big(j, \mu(j)\big)\big) \Big), \qquad \forall\ m.$$

Given V_μ, the improved policy is obtained as

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \Big[g(i,u) + V_\mu\big(f(i,u)\big) \Big], \qquad i = 1, \ldots, n.$$

There are also corresponding approximate policy iteration methods with cost function approximation.

An advantage of this method when implemented by simulation is that the computation of the improved policy does not require the calculation of expected values. Moreover, with a simulator, the policy evaluation of V_μ can be done in model-free fashion, without explicit knowledge of the

probabilities $q(j \mid m)$. These advantages are shared with policy iteration algorithms based on Q-factors. However, when function approximation is used in policy iteration, the methods using post-decision states may have a significant advantage over Q-factor-based methods: they use cost function approximation in the space of post-decision states, rather than the larger space of state-control pairs, and they are less susceptible to difficulties due to inadequate exploration.

We note that there is a similar simplification with post-decision states when g is of the form

$$g(i,u,j) = h\big(f(i,u),j\big),$$

for some function h. Then we have

$$J^*(i) = \min_{u \in U(i)} V^*\big(f(i,u)\big),$$

where V^* is the unique solution of the equation

$$V^*(m) = \sum_{j=1}^{n} q(j \mid m) \left(h(m,j) + \alpha \min_{u \in U(j)} V^*\big(f(j,u)\big) \right), \qquad \forall\ m.$$

Here $V^*(m)$ should be interpreted as the optimal cost-to-go from post-decision state m, *including the cost* $h(m,j)$ *incurred within the stage when* m *was generated*. When h does not depend on j, the algorithm takes the simpler form

$$V^*(m) = h(m) + \alpha \sum_{j=1}^{n} q(j \mid m) \min_{u \in U(j)} V^*\big(f(j,u)\big), \qquad \forall\ m. \tag{6.12}$$

Example 6.1.4 (Tetris)

Let us revisit the game of tetris, which was discussed in Example 1.4.1 of Vol. I in the context of problems with an uncontrollable state component. We will show that it also admits a post-decision state. Assuming that the game terminates with probability 1 for every policy (a proof of this has been given by Burgiel [Bur97]), we can model the problem of finding an optimal tetris playing strategy as a stochastic shortest path problem.

The state consists of two components:

(1) The board position, i.e., a binary description of the full/empty status of each square, denoted by x.

(2) The shape of the current falling block, denoted by y (this is the uncontrollable component).

The control, denoted by u, is the horizontal positioning and rotation applied to the falling block.

Bellman's equation over the space of the controllable state component takes the form

$$\hat{J}(x) = \sum_y p(y) \max_u \Big[g(x,y,u) + \hat{J}\big(f(x,y,u)\big) \Big], \qquad \text{for all } x,$$

where $g(x,y,u)$ and $f(x,y,u)$ are the number of points scored (rows removed), and the board position when the state is (x,y) and control u is applied, respectively [cf. Eq. (6.9)].

This problem also admits a post-decision state. Once u is applied at state (x,y), a new board position m is obtained, and the new state component $\overline{x}$ is obtained from m after removing a number of rows. Thus we have

$$m = f(x,y,u)$$

for some function f, and m also determines the reward of the stage, which has the form $h(m)$ for some m [$h(m)$ is the number of complete rows that can be removed from m]. Thus, m may serve as a post-decision state, and the corresponding Bellman's equation takes the form (6.12), i.e.,

$$V^*(m) = h(m) + \sum_{(\overline{x},\overline{y})}^{n} q(m,\overline{x},\overline{y}) \max_{u \in U(j)} V^*\big(f(\overline{x},\overline{y},u)\big), \qquad \forall\ m,$$

where $(\overline{x},\overline{y})$ is the state that follows m, and $q(m,\overline{x},\overline{y})$ are the corresponding transition probabilities. Note that both of the simplified Bellman's equations share the same characteristic: they involve a deterministic optimization.

Trading off Complexity of Control Space with Complexity of State Space

Suboptimal control using cost function approximation deals fairly well with large state spaces, but still encounters serious difficulties when the number of controls available at each state is large. In particular, the minimization

$$\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \tilde{J}(j,r)\big)$$

using an approximate cost-go function $\tilde{J}(j,r)$ may be very time-consuming. For multistep lookahead schemes, the difficulty is exacerbated, since the required computation grows exponentially with the size of the lookahead horizon. It is thus useful to know that by reformulating the problem, it may be possible to reduce the complexity of the control space by increasing the complexity of the state space. The potential advantage is that the extra state space complexity may still be dealt with by using function approximation and/or rollout.

In particular, suppose that the control u consists of m components,

$$u = (u_1, \ldots, u_m).$$

Then, at a given state i, we can break down u into the sequence of the m controls $u_1, u_2, \ldots, u_m$, and introduce artificial intermediate "states" $(i, u_1), (i, u_1, u_2), \ldots, (i, u_1, \ldots, u_{m-1})$, and corresponding transitions to model the effect of these controls. The choice of the last control component u_m at "state" $(i, u_1, \ldots, u_{m-1})$ marks the transition to state j according to the given transition probabilities $p_{ij}(u)$. In this way the control space is simplified at the expense of introducing $m-1$ additional layers of states, and $m-1$ additional cost-to-go functions

$$J_1(i, u_1), J_2(i, u_1, u_2), \ldots, J_{m-1}(i, u_1, \ldots, u_{m-1}).$$

To deal with the increase in size of the state space we may use rollout, i.e., when at "state" $(i, u_1, \ldots, u_k)$, assume that future controls $u_{k+1}, \ldots, u_m$ will be chosen by a base heuristic. Alternatively, we may use function approximation, that is, introduce cost-to-go approximations

$$\tilde{J}_1(i, u_1, r_1),\ \tilde{J}_2(i, u_1, u_2, r_2), \ldots, \tilde{J}_{m-1}(i, u_1, \ldots, u_{m-1}, r_{m-1}),$$

in addition to $\tilde{J}(i, r)$. We refer to [BeT96], Section 6.1.4, for further discussion.

A potential complication in the preceding schemes arises when the controls $u_1, \ldots, u_m$ are coupled through a constraint of the form

$$u = (u_1, \ldots, u_m) \in U(i). \tag{6.13}$$

Then, when choosing a control u_k, care must be exercised to ensure that the future controls $u_{k+1}, \ldots, u_m$ can be chosen together with the already chosen controls $u_1, \ldots, u_k$ to satisfy the feasibility constraint (6.13). This requires a variant of the rollout algorithm that works with constrained DP problems; see Exercise 6.19 of Vol. I, and also references [Ber05a], [Ber05b].

6.1.5 Monte Carlo Simulation

In this subsection and the next, we will try to provide some orientation into the mathematical content of this chapter. The reader may wish to skip these subsections at first, but return to them later for a higher level view of some of the subsequent technical material.

The methods of this chapter rely to a large extent on simulation in conjunction with cost function approximation in order to deal with large state spaces. The advantage that simulation holds in this regard can be traced to its ability to compute (approximately) sums with a very large number of terms. These sums arise in a number of contexts: inner product and matrix-vector product calculations, the solution of linear systems of equations and policy evaluation, linear least squares problems, etc.

Example 6.1.5 (Approximate Policy Evaluation)

Consider the approximate solution of the Bellman equation that corresponds to a given policy of an n-state discounted problem:

$$J = g + \alpha P J;$$

where P is the transition probability matrix and α is the discount factor. Let us adopt a hard aggregation approach (cf. Section 6.3.4 of Vol. I; see also Section 6.4 later in this chapter), whereby we divide the n states in two disjoint subsets I_1 and I_2 with $I_1 \cup I_2 = \{1, \ldots, n\}$, and we use the piecewise constant approximation

$$J(i) = \begin{cases} r_1 & \text{if } i \in I_1, \\ r_2 & \text{if } i \in I_2. \end{cases}$$

This corresponds to the linear feature-based architecture $J \approx \Phi r$, where Φ is the $n \times 2$ matrix with column components equal to 1 or 0, depending on whether the component corresponds to I_1 or I_2.

We obtain the approximate equations

$$J(i) \approx g(i) + \alpha \left(\sum_{j \in I_1} p_{ij} \right) r_1 + \alpha \left(\sum_{j \in I_2} p_{ij} \right) r_2, \qquad i = 1, \ldots, n,$$

which we can reduce to just two equations by forming two weighted sums (with equal weights) of the equations corresponding to the states in I_1 and I_2, respectively:

$$r_1 \approx \frac{1}{n_1} \sum_{i \in I_1} J(i), \qquad r_2 \approx \frac{1}{n_2} \sum_{i \in I_2} J(i),$$

where n_1 and n_2 are numbers of states in I_1 and I_2, respectively. We thus obtain the aggregate system of the following two equations in r_1 and r_2:

$$r_1 = \frac{1}{n_1} \sum_{i \in I_1} g(i) + \frac{\alpha}{n_1} \left(\sum_{i \in I_1} \sum_{j \in I_1} p_{ij} \right) r_1 + \frac{\alpha}{n_1} \left(\sum_{i \in I_1} \sum_{j \in I_2} p_{ij} \right) r_2,$$

$$r_2 = \frac{1}{n_2} \sum_{i \in I_2} g(i) + \frac{\alpha}{n_2} \left(\sum_{i \in I_2} \sum_{j \in I_1} p_{ij} \right) r_1 + \frac{\alpha}{n_2} \left(\sum_{i \in I_2} \sum_{j \in I_2} p_{ij} \right) r_2.$$

Here the challenge, when the number of states n is very large, is the calculation of the large sums in the right-hand side, which can be of order $O(n^2)$. Simulation allows the approximate calculation of these sums with complexity that is independent of n. This is similar to the advantage that Monte-Carlo integration holds over numerical integration, as discussed in standard texts on Monte-Carlo methods.

To see how simulation can be used with advantage, let us consider the problem of estimating a scalar sum of the form

$$z = \sum_{\omega \in \Omega} v(\omega),$$

where Ω is a finite set and $v : \Omega \mapsto \Re$ is a function of ω. We introduce a distribution ξ that assigns positive probability $\xi(\omega)$ to every element $\omega \in \Omega$ (but is otherwise arbitrary), and we generate a sequence

$$\{\omega_1, \ldots, \omega_T\}$$

of samples from Ω, with each sample ω_t taking values from Ω according to ξ. We then estimate z with

$$\hat{z}_T = \frac{1}{T} \sum_{t=1}^{T} \frac{v(\omega_t)}{\xi(\omega_t)}. \tag{6.14}$$

Clearly $\hat{z}$ is unbiased:

$$E[\hat{z}_T] = \frac{1}{T} \sum_{t=1}^{T} E\left[\frac{v(\omega_t)}{\xi(\omega_t)}\right] = \frac{1}{T} \sum_{t=1}^{T} \sum_{\omega \in \Omega} \xi(\omega) \frac{v(\omega)}{\xi(\omega)} = \sum_{\omega \in \Omega} v(\omega) = z.$$

Suppose now that the samples are generated in a way that the long-term frequency of each $\omega \in \Omega$ is equal to $\xi(\omega)$, i.e.,

$$\lim_{T \to \infty} \sum_{t=1}^{T} \frac{\delta(\omega_t = \omega)}{T} = \xi(\omega), \qquad \forall\, \omega \in \Omega, \tag{6.15}$$

where $\delta(\cdot)$ denotes the indicator function [$\delta(E) = 1$ if the event E has occurred and $\delta(E) = 0$ otherwise]. Then from Eq. (6.14), we have

$$z_T = \sum_{\omega \in \Omega} \sum_{t=1}^{T} \frac{\delta(\omega_t = \omega)}{T} \cdot \frac{v(\omega)}{\xi(\omega)},$$

and by taking limit as $T \to \infty$ and using Eq. (6.15),

$$\lim_{T \to \infty} \hat{z}_T = \sum_{\omega \in \Omega} \lim_{T \to \infty} \sum_{t=1}^{T} \frac{\delta(\omega_t = \omega)}{T} \cdot \frac{v(\omega)}{\xi(\omega)} = \sum_{\omega \in \Omega} v(\omega) = z.$$

Thus in the limit, as the number of samples increases, we obtain the desired sum z. An important case, of particular relevance to the methods of this chapter, is when Ω is the set of states of an irreducible Markov chain. Then, if we generate an infinitely long trajectory $\{\omega_1, \omega_2, \ldots\}$ starting from any

initial state ω_1, then the condition (6.15) will hold with probability 1, with $\xi(\omega)$ being the steady-state probability of state ω.

The samples ω_t need not be independent for the preceding properties to hold, but if they are, then the variance of $\hat{z}_T$ is the sum of the variances of the independent components in the sum of Eq. (6.14), and is given by

$$\mathrm{var}(\hat{z}_T) = \frac{1}{T^2}\sum_{t=1}^{T}\sum_{\omega\in\Omega}\xi(\omega)\left(\frac{v(\omega)}{\xi(\omega)} - z\right)^2 = \frac{1}{T}\sum_{\omega\in\Omega}\xi(\omega)\left(\frac{v(\omega)}{\xi(\omega)} - z\right)^2. \tag{6.16}$$

An important observation from this formula is that the accuracy of the approximation does not depend on the number of terms in the sum z (the number of elements in Ω), but rather depends on the variance of the random variable that takes values $v(\omega)/\xi(\omega)$, $\omega \in \Omega$, with probabilities $\xi(\omega)$.† Thus, it is possible to execute approximately linear algebra operations of very large size through Monte Carlo sampling (with whatever distributions may be convenient in a given context), and this a principal idea underlying the methods of this chapter.

In the case where the samples are dependent, the variance formula (6.16) does not hold, but similar qualitative conclusions can be drawn under various assumptions, which ensure that the dependencies between samples become sufficiently weak over time (see the specialized literature).

Monte Carlo simulation is also important in the context of this chapter for an additional reason. In addition to its ability to compute efficiently sums of very large numbers of terms, it can often do so in model-free fashion (i.e., by using a simulator, rather than an explicit model of the terms in the sum).

6.1.6 Contraction Mappings and Simulation

Most of the chapter (Sections 6.3-6.8) deals with the approximate computation of a fixed point of a (linear or nonlinear) mapping T within a

† The selection of the distribution $\{\xi(\omega) \mid \omega \in \Omega\}$ can be optimized (at least approximately), and methods for doing this are the subject of the technique of *importance sampling*. In particular, assuming that samples are independent and that $v(\omega) \geq 0$ for all $\omega \in \Omega$, we have from Eq. (6.16) that the optimal distribution is $\xi^* = v/z$ and the corresponding minimum variance value is 0. However, ξ^* cannot be computed without knowledge of z. Instead, ξ is usually chosen to be an approximation to v, normalized so that its components add to 1. Note that we may assume that $v(\omega) \geq 0$ for all $\omega \in \Omega$ without loss of generality: when v takes negative values, we may decompose v as

$$v = v^+ - v^-,$$

so that both v^+ and v^- are positive functions, and then estimate separately $z^+ = \sum_{\omega\in\Omega} v^+(\omega)$ and $z^- = \sum_{\omega\in\Omega} v^-(\omega)$.

subspace

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

We will discuss a variety of approaches with distinct characteristics, but at an abstract mathematical level, these approaches fall into two categories:

(a) A *projected equation approach*, based on the equation

$$\Phi r = \Pi T(\Phi r), \tag{6.17}$$

where Π is a projection operation with respect to a Euclidean norm (see Section 6.3 for discounted problems, and Sections 7.1-7.3 for other types of problems).

(b) An *aggregation approach*, based on an equation of the form

$$\Phi r = \Phi DT(\Phi r), \tag{6.18}$$

where D is an $s \times n$ matrix whose rows are probability distributions and Φ are matrices that satisfy certain restrictions.

When iterative methods are used for solution of Eqs. (6.17) and (6.18), it is important that ΠT and ΦDT be contractions over the subspace S. Note here that even if T is a contraction mapping (as is ordinarily the case in DP), it does not follow that ΠT and ΦDT are contractions. In our analysis, this is resolved by requiring that T be a contraction with respect to a norm such that Π or ΦD, respectively, is a nonexpansive mapping. As a result, we need various assumptions on T, Φ, and D, which guide the algorithmic development. We postpone further discussion of these issues, but for the moment we note that the projection approach revolves mostly around Euclidean norm contractions and cases where T is linear, while the aggregation/Q-learning approach revolves mostly around sup-norm contractions.

If T is linear, both equations (6.17) and (6.18) may be written as square systems of linear equations of the form $Cr = d$, whose solution can be approximated by simulation. The approach here is very simple: we approximate C and d with simulation-generated approximations $\hat{C}$ and $\hat{d}$, and we solve the resulting (approximate) linear system $\hat{C}r = \hat{d}$ by matrix inversion, thereby obtaining the solution estimate $\hat{r} = \hat{C}^{-1}\hat{d}$. A primary example is the LSTD methods of Section 6.3.4. We may also try to solve the linear system $\hat{C}r = \hat{d}$ iteratively, which leads to the LSPE type of methods, some of which produce estimates of r simultaneously with the generation of the simulation samples of w (see Section 6.3.4).

Stochastic Approximation Methods

Let us also mention some stochastic iterative algorithms that are based on a somewhat different simulation idea, and fall within the framework of

stochastic approximation methods. The TD(λ) and Q-learning algorithms fall in this category. For an informal orientation, let us consider the computation of the fixed point of a general mapping $F : \Re^n \mapsto \Re^n$ that is a contraction mapping with respect to some norm, and involves an expected value: it has the form

$$F(x) = E\big\{f(x, w)\big\}, \tag{6.19}$$

where $x \in \Re^n$ is a generic argument of F, w is a random variable and $f(\cdot, w)$ is a given function. Assume for simplicity that w takes values in a finite set W with probabilities $p(w)$, so that the fixed point equation $x = F(x)$ has the form

$$x = \sum_{w \in W} p(w) f(x, w).$$

We generate a sequence of samples $\{w_1, w_2, \ldots\}$ such that the empirical frequency of each value $w \in W$ is equal to its probability $p(w)$, i.e.,

$$\lim_{k \to \infty} \frac{n_k(w)}{k} = p(w), \qquad w \in W,$$

where $n_k(w)$ denotes the number of times that w appears in the first k samples $w_1, \ldots, w_k$. This is a reasonable assumption that may be verified by application of various laws of large numbers to the sampling method at hand.

Given the samples, we may consider approximating the fixed point of F by the (approximate) fixed point iteration

$$x_{k+1} = \sum_{w \in W} \frac{n_k(w)}{k} f(x_k, w), \tag{6.20}$$

which can also be equivalently written as

$$x_{k+1} = \frac{1}{k} \sum_{i=1}^{k} f(x_k, w_i). \tag{6.21}$$

We may view Eq. (6.20) as a simulation-based version of the convergent fixed point iteration

$$x_{k+1} = F(x_k) = \sum_{w \in W} p(w) f(x_k, w),$$

where the probabilities $p(w)$ have been replaced by the empirical frequencies $\frac{n_k(w)}{k}$. Thus we expect that the simulation-based iteration (6.21) converges to the fixed point of F.

On the other hand the iteration (6.21) has a major flaw: it requires, for each k, the computation of $f(x_k, w_i)$ for *all* sample values w_i, $i =$

$1, \ldots, k$. An algorithm that requires much less computation than iteration (6.21) is

$$x_{k+1} = \frac{1}{k} \sum_{i=1}^{k} f(x_i, w_i), \qquad k = 1, 2, \ldots, \tag{6.22}$$

where *only one* value of f per sample w_i is computed. This iteration can also be written in the simple recursive form

$$x_{k+1} = (1 - \gamma_k)x_k + \gamma_k f(x_k, w_k), \qquad k = 1, 2, \ldots, \tag{6.23}$$

with the stepsize γ_k having the form $\gamma_k = 1/k$. As an indication of its validity, we note that if it converges to some limit then this limit must be the fixed point of F, since for large k the iteration (6.22) becomes essentially identical to the iteration $x_{k+1} = F(x_k)$. Other stepsize rules, which satisfy $\gamma_k \to 0$ and $\sum_{k=1}^{\infty} \gamma_k = \infty$, may also be used. However, a rigorous analysis of the convergence of iteration (6.23) is nontrivial and is beyond our scope. The book by Bertsekas and Tsitsiklis [BeT96] contains a fairly detailed development, which is tailored to DP. Other more general references are Benveniste, Metivier, and Priouret [BMP90], Borkar [Bor08], Kushner and Yin [KuY03], and Meyn [Mey07].

6.2 DIRECT POLICY EVALUATION - GRADIENT METHODS

We will now consider the direct approach for policy evaluation.† In particular, suppose that the current policy is μ, and for a given r, $\tilde{J}(i, r)$ is an approximation of $J_\mu(i)$. We generate an "improved" policy $\overline{\mu}$ using the formula

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha \tilde{J}(j, r)\big), \qquad \text{for all } i. \tag{6.24}$$

To evaluate approximately $J_{\overline{\mu}}$, we select a subset of "representative" states $\tilde{S}$ (perhaps obtained by some form of simulation), and for each $i \in \tilde{S}$, we obtain $M(i)$ samples of the cost $J_{\overline{\mu}}(i)$. The mth such sample is denoted by

† Direct policy evaluation methods have been historically important, and provide an interesting contrast with indirect methods. However, they are currently less popular than the projected equation methods to be considered in the next section, despite some generic advantages (the option to use nonlinear approximation architectures, and the capability of more accurate approximation). The material of this section will not be substantially used later, so the reader may read lightly this section without loss of continuity.

$c(i,m)$, and mathematically, it can be viewed as being $J_{\overline{\mu}}(i)$ plus some simulation error/noise.‡ Then we obtain the corresponding parameter vector $\overline{r}$ by solving the following least squares problem

$$\min_{\overline{r}} \sum_{i \in \tilde{S}} \sum_{m=1}^{M(i)} \big(\tilde{J}(i,\overline{r}) - c(i,m)\big)^2, \tag{6.25}$$

and we repeat the process with $\overline{\mu}$ and $\overline{r}$ replacing μ and r, respectively (see Fig. 6.1.1).

The least squares problem (6.25) can be solved exactly if a linear approximation architecture is used, i.e., if

$$\tilde{J}(i,\overline{r}) = \phi(i)'\overline{r},$$

where $\phi(i)'$ is a row vector of features corresponding to state i. In this case $\overline{r}$ is obtained by solving the linear system of equations

$$\sum_{i \in \tilde{S}} \sum_{m=1}^{M(i)} \phi(i)\big(\phi(i)'\overline{r} - c(i,m)\big) = 0,$$

which is obtained by setting to 0 the gradient with respect to $\overline{r}$ of the quadratic cost in the minimization (6.25). When a nonlinear architecture is used, we may use gradient-like methods for solving the least squares problem (6.25), as we will now discuss.

Batch Gradient Methods for Policy Evaluation

Let us focus on an N-transition portion $(i_0,\ldots,i_N)$ of a simulated trajectory, also called a *batch*. We view the numbers

$$\sum_{t=k}^{N-1} \alpha^{t-k} g\big(i_t, \overline{\mu}(i_t), i_{t+1}\big), \qquad k = 0,\ldots,N-1,$$

‡ The manner in which the samples $c(i,m)$ are collected is immaterial for the purposes of the subsequent discussion. Thus one may generate these samples through a single very long trajectory of the Markov chain corresponding to μ, or one may use multiple trajectories, with different starting points, to ensure that enough cost samples are generated for a "representative" subset of states. In either case, the samples $c(i,m)$ corresponding to any one state i will generally be correlated as well as "noisy." Still the average $\frac{1}{M(i)}\sum_{m=1}^{M(i)} c(i,m)$ will ordinarily converge to $J_\mu(i)$ as $M(i) \to \infty$ by a law of large numbers argument [see Exercise 6.2 and the discussion in [BeT96], Sections 5.1, 5.2, regarding the behavior of the average when $M(i)$ is finite and random].

as cost samples, one per initial state $i_0, \ldots, i_{N-1}$, which can be used for least squares approximation of the parametric architecture $\tilde{J}(i, \overline{r})$ [cf. Eq. (6.25)]:

$$\min_{\overline{r}} \sum_{k=0}^{N-1} \frac{1}{2} \left(\tilde{J}(i_k, \overline{r}) - \sum_{t=k}^{N-1} \alpha^{t-k} g\big(i_t, \overline{\mu}(i_t), i_{t+1}\big) \right)^2. \tag{6.26}$$

One way to solve this least squares problem is to use a gradient method, whereby the parameter $\overline{r}$ associated with $\overline{\mu}$ is updated at time N by

$$\overline{r} := \overline{r} - \gamma \sum_{k=0}^{N-1} \nabla \tilde{J}(i_k, \overline{r}) \left(\tilde{J}(i_k, \overline{r}) - \sum_{t=k}^{N-1} \alpha^{t-k} g\big(i_t, \overline{\mu}(i_t), i_{t+1}\big) \right). \tag{6.27}$$

Here, $\nabla \tilde{J}$ denotes gradient with respect to $\overline{r}$ and γ is a positive stepsize, which is usually diminishing over time (we leave its precise choice open for the moment). Each of the N terms in the summation in the right-hand side above is the gradient of a corresponding term in the least squares summation of problem (6.26). Note that the update of $\overline{r}$ is done after processing the entire batch, and that the gradients $\nabla \tilde{J}(i_k, \overline{r})$ are evaluated at the preexisting value of $\overline{r}$, i.e., the one before the update.

In a traditional gradient method, the gradient iteration (6.27) is repeated, until convergence to the solution of the least squares problem (6.26), i.e., a single N-transition batch is used. However, there is an important tradeoff relating to the size N of the batch: in order to reduce simulation error and generate multiple cost samples for a representatively large subset of states, it is necessary to use a large N, yet to keep the work per gradient iteration small it is necessary to use a small N.

To address the issue of size of N, an expanded view of the gradient method is preferable in practice, whereby batches may be changed after one or more iterations. Thus, in this more general method, the N-transition batch used in a given gradient iteration comes from a potentially longer simulated trajectory, or from one of many simulated trajectories. A sequence of gradient iterations is performed, with each iteration using cost samples formed from batches collected in a variety of different ways and whose length N may vary. Batches may also overlap to a substantial degree.

We leave the method for generating simulated trajectories and forming batches open for the moment, but we note that it influences strongly the result of the corresponding least squares optimization (6.25), providing better approximations for the states that arise most frequently in the batches used. This is related to the issue of ensuring that the state space is adequately "explored," with an adequately broad selection of states being represented in the least squares optimization, cf. our earlier discussion on the exploration issue.

The gradient method (6.27) is simple, widely known, and easily understood. There are extensive convergence analyses of this method and

its variations, for which we refer to the literature cited at the end of the chapter. These analyses often involve considerable mathematical sophistication, particularly when multiple batches are involved, because of the stochastic nature of the simulation and the complex correlations between the cost samples. However, qualitatively, the conclusions of these analyses are consistent among themselves as well as with practical experience, and indicate that:

(1) Under some reasonable technical assumptions, convergence to a limiting value of $\overline{r}$ that is a local minimum of the associated optimization problem is expected.

(2) For convergence, it is essential to gradually reduce the stepsize to 0, the most popular choice being to use a stepsize proportional to $1/m$, while processing the mth batch. In practice, considerable trial and error may be needed to settle on an effective stepsize choice method. Sometimes it is possible to improve performance by using a different stepsize (or scaling factor) for each component of the gradient.

(3) The rate of convergence is often very slow, and depends among other things on the initial choice of $\overline{r}$, the number of states and the dynamics of the associated Markov chain, the level of simulation error, and the method for stepsize choice. In fact, the rate of convergence is sometimes so slow, that practical convergence is infeasible, even if theoretical convergence is guaranteed.

Incremental Gradient Methods for Policy Evaluation

We will now consider a variant of the gradient method called *incremental*. This method can also be described through the use of N-transition batches, but we will see that (contrary to the batch version discussed earlier) the method is suitable for use with very long batches, including the possibility of a single very long simulated trajectory, viewed as a single batch.

For a given N-transition batch $(i_0, \ldots, i_N)$, the batch gradient method processes the N transitions all at once, and updates $\overline{r}$ using Eq. (6.27). The incremental method updates $\overline{r}$ a total of N times, *once after each transition*. Each time it adds to $\overline{r}$ the corresponding portion of the gradient in the right-hand side of Eq. (6.27) that can be calculated using the newly available simulation data. Thus, after each transition (i_k, i_{k+1}):

(1) We evaluate the gradient $\nabla \tilde{J}(i_k, \overline{r})$ at the current value of $\overline{r}$.

(2) We sum all the terms in the right-hand side of Eq. (6.27) that involve the transition (i_k, i_{k+1}), and we update $\overline{r}$ by making a correction

along their sum:

$$\overline{r} := \overline{r} - \gamma \left(\nabla \tilde{J}(i_k, \overline{r}) \tilde{J}(i_k, \overline{r}) - \left(\sum_{t=0}^{k} \alpha^{k-t} \nabla \tilde{J}(i_t, \overline{r}) \right) g\big(i_k, \overline{\mu}(i_k), i_{k+1}\big) \right). \tag{6.28}$$

By adding the parenthesized "incremental" correction terms in the above iteration, we see that after N transitions, all the terms of the batch iteration (6.27) will have been accumulated, but there is a difference: in the incremental version, $\overline{r}$ is changed during the processing of the batch, and the gradient $\nabla \tilde{J}(i_t, \overline{r})$ is evaluated at the most recent value of $\overline{r}$ [after the transition (i_t, i_{t+1})]. By contrast, in the batch version these gradients are evaluated at the value of $\overline{r}$ prevailing at the beginning of the batch. Note that the gradient sum in the right-hand side of Eq. (6.28) can be conveniently updated following each transition, thereby resulting in an efficient implementation.

It can now be seen that because $\overline{r}$ is updated at intermediate transitions within a batch (rather than at the end of the batch), the location of the end of the batch becomes less relevant. It is thus possible to have very long batches, and indeed the algorithm can be operated with a single very long simulated trajectory and a single batch. In this case, for each state i, we will have one cost sample for every time when state i is encountered in the simulation. Accordingly state i will be weighted in the least squares optimization in proportion to the frequency of its occurrence within the simulated trajectory.

Generally, within the least squares/policy evaluation context of this section, the incremental versions of the gradient methods can be implemented more flexibly and tend to converge faster than their batch counterparts, so they will be adopted as the default in our discussion. The book by Bertsekas and Tsitsiklis [BeT96] contains an extensive analysis of the theoretical convergence properties of incremental gradient methods (they are fairly similar to those of batch methods), and provides some insight into the reasons for their superior performance relative to the batch versions; see also the author's nonlinear programming book [Ber99] (Section 1.5.2), the paper by Bertsekas and Tsitsiklis [BeT00], and the author's recent survey [Ber10d]. Still, however, the rate of convergence can be very slow.

Implementation Using Temporal Differences – TD(1)

We now introduce an alternative, mathematically equivalent, implementation of the batch and incremental gradient iterations (6.27) and (6.28), which is described with cleaner formulas. It uses the notion of *temporal difference* (TD for short) given by

$$q_k = \tilde{J}(i_k, \overline{r}) - \alpha \tilde{J}(i_{k+1}, \overline{r}) - g\big(i_k, \overline{\mu}(i_k), i_{k+1}\big), \qquad k = 0, \ldots, N-2, \tag{6.29}$$

$$q_{N-1} = \tilde{J}(i_{N-1}, \overline{r}) - g\big(i_{N-1}, \overline{\mu}(i_{N-1}), i_N\big). \tag{6.30}$$

In particular, by noting that the parenthesized term multiplying $\nabla \tilde{J}(i_k, \overline{r})$ in Eq. (6.27) is equal to

$$q_k + \alpha q_{k+1} + \cdots + \alpha^{N-1-k} q_{N-1},$$

we can verify by adding the equations below that iteration (6.27) can also be implemented as follows:

After the state transition (i_0, i_1), set

$$\overline{r} := \overline{r} - \gamma q_0 \nabla \tilde{J}(i_0, \overline{r}).$$

After the state transition (i_1, i_2), set

$$\overline{r} := \overline{r} - \gamma q_1 \big(\alpha \nabla \tilde{J}(i_0, \overline{r}) + \nabla \tilde{J}(i_1, \overline{r})\big).$$

Proceeding similarly, after the state transition (i_{N-1}, t), set

$$\overline{r} := \overline{r} - \gamma q_{N-1} \big(\alpha^{N-1} \nabla \tilde{J}(i_0, \overline{r}) + \alpha^{N-2} \nabla \tilde{J}(i_1, \overline{r}) + \cdots + \nabla \tilde{J}(i_{N-1}, \overline{r})\big).$$

The batch version (6.27) is obtained if the gradients $\nabla \tilde{J}(i_k, \overline{r})$ are all evaluated at the value of $\overline{r}$ that prevails at the beginning of the batch. The incremental version (6.28) is obtained if each gradient $\nabla \tilde{J}(i_k, \overline{r})$ is evaluated at the value of $\overline{r}$ that prevails when the transition (i_k, i_{k+1}) is processed.

In particular, for the incremental version, we start with some vector r_0, and following the transition (i_k, i_{k+1}), $k = 0, \ldots, N-1$, we set

$$r_{k+1} = r_k - \gamma_k q_k \sum_{t=0}^{k} \alpha^{k-t} \nabla \tilde{J}(i_t, r_t), \tag{6.31}$$

where the stepsize γ_k may very from one transition to the next. In the important case of a linear approximation architecture of the form

$$\tilde{J}(i, r) = \phi(i)' r, \qquad i = 1, \ldots, n,$$

where $\phi(i) \in \Re^s$ are some fixed vectors, it takes the form

$$r_{k+1} = r_k - \gamma_k q_k \sum_{t=0}^{k} \alpha^{k-t} \phi(i_t). \tag{6.32}$$

This algorithm is known as TD(1), and we will see in Section 6.3.6 that it is a limiting version (as $\lambda \to 1$) of the TD(λ) method discussed there.

6.3 PROJECTED EQUATION METHODS

In this section, we consider the indirect approach, whereby the policy evaluation is based on solving a projected form of Bellman's equation (cf. the right-hand side of Fig. 6.1.5). We will be dealing with a single stationary policy μ, so we generally suppress in our notation the dependence on control of the transition probabilities and the cost per stage. We thus consider a stationary finite-state Markov chain, and we denote the states by $i = 1, \ldots, n$, the transition probabilities by p_{ij}, $i, j = 1, \ldots, n$, and the stage costs by $g(i, j)$. We want to evaluate the expected cost of μ corresponding to each initial state i, given by

$$J_\mu(i) = \lim_{N\to\infty} E\left\{ \sum_{k=0}^{N-1} \alpha^k g(i_k, i_{k+1}) \;\middle|\; i_0 = i \right\}, \qquad i = 1, \ldots, n,$$

where i_k denotes the state at time k, and $\alpha \in (0, 1)$ is the discount factor.

We approximate $J_\mu(i)$ with a linear architecture of the form

$$\tilde{J}(i, r) = \phi(i)'r, \qquad i = 1, \ldots, n, \tag{6.33}$$

where r is a parameter vector and $\phi(i)$ is an s-dimensional feature vector associated with the state i. (Throughout this section, vectors are viewed as column vectors, and a prime denotes transposition.) As earlier, we also write the vector

$$\big(\tilde{J}(1, r), \ldots, \tilde{J}(n, r)\big)'$$

in the compact form Φr, where Φ is the $n \times s$ matrix that has as rows the feature vectors $\phi(i)$, $i = 1, \ldots, n$. Thus, we want to approximate J_μ within

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

the subspace spanned by s basis functions, the columns of Φ. Our assumptions in this section are the following (we will later discuss how our methodology may be modified in the absence of these assumptions).

Assumption 6.3.1: The Markov chain has steady-state probabilities $\xi_1, \ldots, \xi_n$, which are positive, i.e., for all $i = 1, \ldots, n$,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} P(i_k = j \mid i_0 = i) = \xi_j > 0, \qquad j = 1, \ldots, n.$$

Assumption 6.3.2: The matrix Φ has rank s.

Assumption 6.3.1 is equivalent to assuming that the Markov chain is *irreducible*, i.e., has a single recurrent class and no transient states. Assumption 6.3.2 is equivalent to the basis functions (the columns of Φ) being linearly independent, and is analytically convenient because it implies that each vector J in the subspace S is represented in the form Φr with a unique vector r.

6.3.1 The Projected Bellman Equation

We will now introduce the projected form of Bellman's equation. We use a weighted Euclidean norm on $\Re^n$ of the form

$$\|J\|_v = \sqrt{\sum_{i=1}^{n} v_i \big(J(i)\big)^2},$$

where v is a vector of positive weights $v_1, \ldots, v_n$. Let Π denote the projection operation onto S with respect to this norm. Thus for any $J \in \Re^n$, ΠJ is the unique vector in S that minimizes $\|J - \hat{J}\|_v^2$ over all $\hat{J} \in S$. It can also be written as

$$\Pi J = \Phi r_J,$$

where

$$r_J = \arg\min_{r \in \Re^s} \|J - \Phi r\|_v^2, \qquad J \in \Re^n. \tag{6.34}$$

This is because Φ has rank s by Assumption 6.3.2, so a vector in S is uniquely written in the form Φr.

Note that Π and r_J can be written explicitly in closed form. This can be done by setting to 0 the gradient of the quadratic function

$$\|J - \Phi r\|_v^2 = (J - \Phi r)' V (J - \Phi r),$$

where V is the diagonal matrix with v_i, $i = 1, \ldots, n$, along the diagonal [cf. Eq. (6.34)]. We thus obtain the necessary and sufficient optimality condition

$$\Phi' V (J - \Phi r_J) = 0, \tag{6.35}$$

from which

$$r_J = (\Phi' V \Phi)^{-1} \Phi' V J,$$

and using the formula $\Phi r_J = \Pi J$,

$$\Pi = \Phi (\Phi' V \Phi)^{-1} \Phi' V.$$

[The inverse $(\Phi' V \Phi)^{-1}$ exists because Φ is assumed to have rank s; cf. Assumption 6.3.2.] The optimality condition (6.35), through left multiplication with r', can also be equivalently expressed as

$$(\Phi r)' V (J - \Phi r_J) = 0, \qquad \forall\ \Phi r \in S. \tag{6.36}$$

The interpretation is that *the difference/approximation error $J - \Phi r_J$ is orthogonal to the subspace S in the scaled geometry of the norm $\|\cdot\|_v$* (two vectors $x, y \in \Re^n$ are called orthogonal if $x'Vy = \sum_{i=1}^n v_i x_i y_i = 0$).

Consider now the mapping T given by

$$(TJ)(i) = \sum_{j=1}^n p_{ij}\big(g(i,j) + \alpha J(j)\big), \qquad i = 1, \ldots, n,$$

the mapping ΠT (the composition of Π with T), and the equation

$$\Phi r = \Pi T(\Phi r). \tag{6.37}$$

We view this as a projected/approximate form of Bellman's equation, and we view a solution Φr^* of this equation as an approximation to J_μ. Note that Φr^* depends only on the projection norm and the subspace S, and not on the matrix Φ, which provides just an algebraic representation of S, i.e., all matrices Φ whose range space is S result in identical vectors Φr^*).

We know from Section 1.4 that T is a contraction with respect to the sup-norm, but unfortunately this does not necessarily imply that T is a contraction with respect to the norm $\|\cdot\|_v$. We will next show an important fact: if v is chosen to be the steady-state probability vector ξ, then T is a contraction with respect to $\|\cdot\|_v$, with modulus α. The critical part of the proof is addressed in the following lemma.

Lemma 6.3.1: For any $n \times n$ stochastic matrix P that has a steady-state probability vector $\xi = (\xi_1, \ldots, \xi_n)$ with positive components, we have

$$\|Pz\|_\xi \le \|z\|_\xi, \qquad z \in \Re^n.$$

Proof: Let p_{ij} be the components of P. For all $z \in \Re^n$, we have

$$\begin{aligned}
\|Pz\|_\xi^2 &= \sum_{i=1}^n \xi_i \left(\sum_{j=1}^n p_{ij} z_j \right)^2 \\
&\le \sum_{i=1}^n \xi_i \sum_{j=1}^n p_{ij} z_j^2 \\
&= \sum_{j=1}^n \sum_{i=1}^n \xi_i p_{ij} z_j^2 \\
&= \sum_{j=1}^n \xi_j z_j^2 \\
&= \|z\|_\xi^2,
\end{aligned}$$

where the inequality follows from the convexity of the quadratic function, and the next to last equality follows from the defining property $\sum_{i=1}^{n} \xi_i p_{ij} = \xi_j$ of the steady-state probabilities. **Q.E.D.**

We next note an important property of projections: they are *nonexpansive*, in the sense

$$\|\Pi J - \Pi \bar{J}\|_v \leq \|J - \bar{J}\|_v, \qquad \text{for all } J, \bar{J} \in \Re^n.$$

To see this, note that by using the linearity of Π, we have

$$\|\Pi J - \Pi \bar{J}\|_v^2 = \big\|\Pi(J - \bar{J})\big\|_v^2 \leq \big\|\Pi(J - \bar{J})\big\|_v^2 + \big\|(I - \Pi)(J - \bar{J})\big\|_v^2 = \|J - \bar{J}\|_v^2,$$

where the rightmost equality follows from the Pythagorean Theorem:†

$$\|X\|_v^2 = \|\Pi X\|_v^2 + \|(I - \Pi)X\|_v^2, \qquad \text{for all } X \in \Re^n, \tag{6.38}$$

applied with $X = J - \bar{J}$. Thus, for ΠT to be a contraction with respect to $\|\cdot\|_v$, it is sufficient that T be a contraction with respect to $\|\cdot\|_v$, since

$$\|\Pi T J - \Pi T \bar{J}\|_v \leq \|T J - T \bar{J}\|_v \leq \beta \|J - \bar{J}\|_v,$$

where β is the modulus of contraction of T with respect to $\|\cdot\|_v$ (see Fig. 6.3.1). This leads to the following proposition.

Proposition 6.3.1: The mappings T and ΠT are contractions of modulus α with respect to the weighted Euclidean norm $\|\cdot\|_\xi$, where ξ is the steady-state probability vector of the Markov chain.

Proof: We write T in the form $TJ = g + \alpha P J$, where g is the vector with components $\sum_{j=1}^{n} p_{ij} g(i,j)$, $i = 1, \ldots, n$, and P is the matrix with components p_{ij}. Then we have for all $J, \bar{J} \in \Re^n$,

$$TJ - T\bar{J} = \alpha P (J - \bar{J}).$$

We thus obtain

$$\|TJ - T\bar{J}\|_\xi = \alpha \|P(J - \bar{J})\|_\xi \leq \alpha \|J - \bar{J}\|_\xi,$$

where the inequality follows from Lemma 6.3.1. Hence T is a contraction of modulus α. The contraction property of ΠT follows from the contraction property of T and the nonexpansiveness property of Π noted earlier. **Q.E.D.**

† The Pythagorean Theorem follows from the orthogonality of the vectors ΠX and $(I - \Pi)X$ in the scaled geometry of the norm $\|\cdot\|_v$.

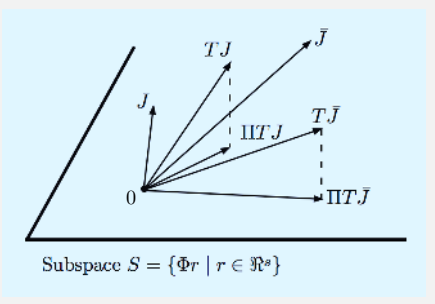

Figure 6.3.1 Illustration of the contraction property of ΠT due to the nonexpansiveness of Π. If T is a contraction with respect to $\|\cdot\|_v$, the Euclidean norm used in the projection, then ΠT is also a contraction with respect to that norm, since Π is nonexpansive and we have

$$\|\Pi TJ - \Pi T\bar{J}\|_v \leq \|TJ - T\bar{J}\|_v \leq \beta\|J - \bar{J}\|_v,$$

where β is the modulus of contraction of T with respect to $\|\cdot\|_v$.

The next proposition gives an estimate of the error in estimating J_μ with the fixed point of ΠT.

Proposition 6.3.2: Let Φr^* be the fixed point of ΠT. We have

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha^2}}\,\|J_\mu - \Pi J_\mu\|_\xi.$$

Proof: We have

$$\begin{aligned}\|J_\mu - \Phi r^*\|_\xi^2 &= \|J_\mu - \Pi J_\mu\|_\xi^2 + \left\|\Pi J_\mu - \Phi r^*\right\|_\xi^2 \\ &= \|J_\mu - \Pi J_\mu\|_\xi^2 + \left\|\Pi TJ_\mu - \Pi T(\Phi r^*)\right\|_\xi^2 \\ &\leq \|J_\mu - \Pi J_\mu\|_\xi^2 + \alpha^2\|J_\mu - \Phi r^*\|_\xi^2,\end{aligned}$$

where the first equality uses the Pythagorean Theorem [cf. Eq. (6.38) with $X = J_\mu - \Phi r^*$], the second equality holds because J_μ is the fixed point of T and Φr^* is the fixed point of ΠT, and the inequality uses the contraction property of ΠT. From this relation, the result follows. **Q.E.D.**

Note the critical fact in the preceding analysis: αP (and hence T) is a contraction with respect to the projection norm $\|\cdot\|_\xi$ (cf. Lemma 6.3.1). Indeed, Props. 6.3.1 and 6.3.2 hold if T is any (possibly nonlinear)

contraction with respect to the Euclidean norm of the projection (cf. Fig. 6.3.1).

The Matrix Form of the Projected Bellman Equation

Let us now write the projected Bellman equation $\Phi r = \Pi T(\Phi r)$ in explicit form. We note that this is a linear equation, since the projection Π is linear and also T is linear of the form

$$TJ = g + \alpha PJ,$$

where g is the vector with components $\sum_{j=1}^{n} p_{ij}g(i,j)$, $i = 1, \ldots, n$, and P is the matrix with components p_{ij}. The solution of the projected Bellman equation is the vector $J = \Phi r^*$, where r^* satisfies the orthogonality condition

$$\Phi'\Xi\big(\Phi r^* - (g + \alpha P\Phi r^*)\big) = 0, \tag{6.39}$$

with Ξ being the diagonal matrix with the steady-state probabilities $\xi_1, \ldots, \xi_n$ along the diagonal [cf. Eq. (6.36)].†

Thus the projected equation is written as

$$Cr^* = d, \tag{6.40}$$

where

$$C = \Phi'\Xi(I - \alpha P)\Phi, \qquad d = \Phi'\Xi g, \tag{6.41}$$

and can be solved by matrix inversion:

$$r^* = C^{-1}d,$$

just like the Bellman equation, which can also be solved by matrix inversion,

$$J = (I - \alpha P)^{-1}g.$$

An important difference is that the projected equation has smaller dimension (s rather than n). Still, however, computing C and d using Eq. (6.41), requires computation of inner products of size n, so for problems where n is very large, the explicit computation of C and d is impractical. We will discuss shortly efficient methods to compute inner products of large size by using simulation and low dimensional calculations. The idea is that an inner product, appropriately normalized, can be viewed as an expected value (the weighted sum of a large number of terms), which can be computed by sampling its components with an appropriate probability distribution and averaging the samples, as discussed in Section 6.1.5.

† Here Φr^* is the projection of $g+\alpha P\Phi r^*$, so $\Phi r^* - (g+\alpha P\Phi r^*)$ is orthogonal to the columns of Φ. Alternatively, r^* solves the problem

$$\min_{r\in\Re^s} \big\|\Phi r - (g + \alpha P\Phi r^*)\big\|_\xi^2.$$

Setting to 0 the gradient with respect to r of the above quadratic expression, we obtain Eq. (6.39).

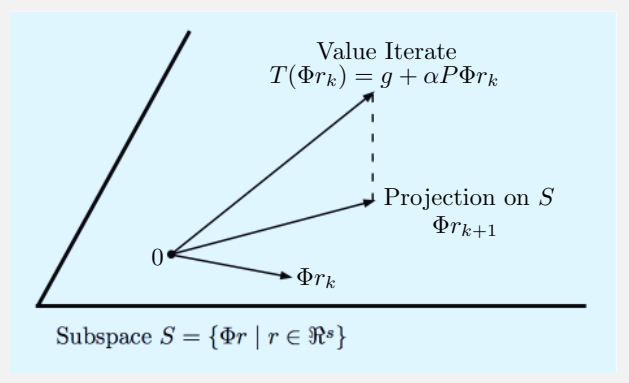

Figure 6.3.2 Illustration of the projected value iteration (PVI) method

$$\Phi r_{k+1} = \Pi T(\Phi r_k).$$

At the typical iteration k, the current iterate Φr_k is operated on with T, and the generated vector $T(\Phi r_k)$ is projected onto S, to yield the new iterate Φr_{k+1}.

6.3.2 Projected Value Iteration - Other Iterative Methods

We noted in Chapter 1 that for problems where n is very large, an iterative method such as value iteration may be appropriate for solving the Bellman equation $J = TJ$. Similarly, one may consider an iterative method for solving the projected Bellman equation $\Phi r = \Pi T(\Phi r)$ or its equivalent version $Cr = d$ [cf. Eqs. (6.40)-(6.41)].

Since ΠT is a contraction (cf. Prop. 6.3.1), the first iterative method that comes to mind is the analog of value iteration: successively apply ΠT, starting with an arbitrary initial vector Φr_0:

$$\Phi r_{k+1} = \Pi T(\Phi r_k), \qquad k = 0, 1, \ldots. \tag{6.42}$$

Thus at iteration k, the current iterate Φr_k is operated on with T, and the generated value iterate $T(\Phi r_k)$ (which does not necessarily lie in S) is projected onto S, to yield the new iterate Φr_{k+1} (see Fig. 6.3.2). We refer to this as *projected value iteration* (PVI for short). Since ΠT is a contraction, it follows that the sequence $\{\Phi r_k\}$ generated by PVI converges to the unique fixed point Φr^* of ΠT.

It is possible to write PVI explicitly by noting that

$$r_{k+1} = \arg\min_{r \in \Re^s} \big\|\Phi r - (g + \alpha P \Phi r_k)\big\|_\xi^2.$$

By setting to 0 the gradient with respect to r of the above quadratic expression, we obtain the orthogonality condition

$$\Phi' \Xi \big(\Phi r_{k+1} - (g + \alpha P \Phi r_k)\big) = 0,$$

[cf. Eq. (6.39)], which yields

$$r_{k+1} = r_k - (\Phi'\Xi\Phi)^{-1}(Cr_k - d), \tag{6.43}$$

where C and d are given by Eq. (6.41).

From the point of view of DP, the PVI method makes intuitive sense, and connects well with established DP theory. However, the methodology of iterative methods for solving linear equations suggests a much broader set of algorithmic possibilities. In particular, in a generic lass of methods, the current iterate r_k is corrected by the "residual" $Cr_k - d$ (which tends to 0), after "scaling" with some $s \times s$ scaling matrix G, leading to the iteration

$$r_{k+1} = r_k - \gamma G(Cr_k - d), \tag{6.44}$$

where γ is a positive stepsize, and G is some $s \times s$ scaling matrix.† When $G = (\Phi'\Xi\Phi)^{-1}$ and $\gamma = 1$, we obtain the PVI method, but there are other interesting possibilities. For example when G is the identity or a diagonal approximation to $(\Phi'\Xi\Phi)^{-1}$, the iteration (6.44) is simpler than PVI in that it does not require a matrix inversion (it does require, however, the choice of a stepsize γ).

The iteration (6.44) converges to the solution of the projected equation if and only if the matrix $I - \gamma GC$ has eigenvalues strictly within the unit circle. The following proposition shows that this is true when G is positive definite symmetric, as long as the stepsize γ is small enough to compensate for large components in the matrix G. This hinges on an important property of the matrix C, which we now define. Let us say that a (possibly nonsymmetric) $s \times s$ matrix M is *positive definite* if

$$r'Mr > 0, \qquad \forall\ r \neq 0.$$

We say that M is *positive semidefinite* if

$$r'Mr \geq 0, \qquad \forall\ r \in \Re^s.$$

The following proposition shows that C is positive definite, and if G is positive definite and symmetric, the iteration (6.44) is convergent for sufficiently small stepsize γ.

† Iterative methods that involve incremental changes along directions of the form $Gf(x)$ are very common for solving a system of equations $f(x) = 0$. They arise prominently in cases where $f(x)$ is the gradient of a cost function, or has certain monotonicity properties. They also admit extensions to the case where there are constraints on x (see [Ber09b], [Ber11a] for an analysis that is relevant to the present DP context).

Proposition 6.3.3: The matrix C of Eq. (6.41) is positive definite. Furthermore, if the $s \times s$ matrix G is symmetric and positive definite, there exists $\overline{\gamma} > 0$ such that the eigenvalues of

$$I - \gamma GC$$

lie strictly within the unit circle for all $\gamma \in (0, \overline{\gamma}]$.

For the proof we need the following lemma, which is attributed to Lyapunov (see Theorem 3.3.9, and Note 3.13.6 of Cottle, Pang, and Stone [CPS92]).

Lemma 6.3.2: The eigenvalues of a positive definite matrix have positive real parts.

Proof: Let M be a positive definite matrix. Then for sufficiently small $\gamma > 0$ we have $(\gamma/2)r'M'Mr < r'Mr$ for all $r \neq 0$, or equivalently

$$\big\|(I - \gamma M)r\big\|^2 < \|r\|^2, \qquad \forall\ r \neq 0,$$

implying that $I - \gamma M$ is a contraction mapping with respect to the standard Euclidean norm. Hence the eigenvalues of $I - \gamma M$ lie within the unit circle. Since these eigenvalues are $1 - \gamma\lambda$, where λ are the eigenvalues of M, it follows that if M is positive definite, the eigenvalues of M have positive real parts. **Q.E.D.**

Proof of Prop. 6.3.3: For all $r \in \Re^s$, we have

$$\|\Pi P\Phi r\|_\xi \leq \|P\Phi r\|_\xi \leq \|\Phi r\|_\xi, \tag{6.45}$$

where the first inequality follows from the Pythagorean Theorem,

$$\|P\Phi r\|_\xi^2 = \|\Pi P\Phi r\|_\xi^2 + \|(I - \Pi)P\Phi r\|_\xi^2,$$

and the second inequality follows from Prop. 6.3.1. Also from properties of projections, all vectors of the form Φr are orthogonal to all vectors of the form $x - \Pi x$, i.e.,

$$r'\Phi'\Xi(I - \Pi)x = 0, \qquad \forall\ r \in \Re^s,\ x \in \Re^n, \tag{6.46}$$

[cf. Eq. (6.36)]. Thus, we have for all $r \neq 0$,

$$r'Cr = r'\Phi'\Xi(I - \alpha P)\Phi r$$

$$
\begin{aligned}
&= r'\Phi'\Xi\big(I - \alpha\Pi P + \alpha(\Pi - I)P\big)\Phi r \\
&= r'\Phi'\Xi(I - \alpha\Pi P)\Phi r \\
&= \|\Phi r\|_\xi^2 - \alpha r'\Phi'\Xi\Pi P\Phi r \\
&\geq \|\Phi r\|_\xi^2 - \alpha\|\Phi r\|_\xi \cdot \|\Pi P\Phi r\|_\xi \\
&\geq (1-\alpha)\|\Phi r\|_\xi^2 \\
&> 0,
\end{aligned}
$$

where the third equality follows from Eq. (6.46), the first inequality follows from the Cauchy-Schwartz inequality applied with inner product $< x, y >= x'\Xi y$, and the second inequality follows from Eq. (6.45). This proves the positive definiteness of C.

If G is symmetric and positive definite, the matrix $G^{1/2}$ exists and is symmetric and positive definite. Let $M = G^{1/2}CG^{1/2}$, and note that since C is positive definite, M is also positive definite, so from Lemma 6.3.2 it follows that its eigenvalues have positive real parts. The eigenvalues of M and GC are equal (with eigenvectors that are multiples of $G^{1/2}$ or $G^{-1/2}$ of each other), so the eigenvalues of GC have positive real parts. It follows that the eigenvalues of $I-\gamma GC$ lie strictly within the unit circle for sufficiently small $\gamma > 0$. This completes the proof of Prop. 6.3.3. **Q.E.D.**

Note that for the conclusion of Prop. 6.3.3 to hold, it is not necessary that G is symmetric. It is sufficient that GC has eigenvalues with positive real parts. An example is $G = C'\Sigma^{-1}$, where Σ is a positive definite symmetric matrix, in which case $GC = C'\Sigma^{-1}C$ is a positive definite matrix. Another example, which is important for our purposes as we will see later (cf., Section 6.3.4), is

$$G = (C'\Sigma^{-1}C + \beta I)^{-1}C'\Sigma^{-1}, \tag{6.47}$$

where Σ is a positive definite symmetric matrix, and β is a positive scalar. Then GC is given by

$$GC = (C'\Sigma^{-1}C + \beta I)^{-1}C'\Sigma^{-1}C,$$

and can be shown to have real eigenvalues that lie in the interval $(0, 1)$, even if C is not positive definite.† As a result $I-\gamma GC$ has real eigenvalues in the interval $(0, 1)$ for any $\gamma \in (0, 2]$.

† To see this let $\lambda_1, \ldots, \lambda_s$ be the eigenvalues of $C'\Sigma^{-1}C$ and let $U\Lambda U'$ be its singular value decomposition, where $\Lambda = \text{diag}\{\lambda_1, \ldots, \lambda_s\}$ and U is a unitary matrix ($UU' = I$; see [Str09], [TrB97]). We also have $C'\Sigma^{-1}C + \beta I = U(\Lambda + \beta I)U'$, so

$$GC = \big(U(\Lambda + \beta I)U'\big)^{-1}U\Lambda U' = U(\Lambda + \beta I)^{-1}\Lambda U'.$$

It follows that the eigenvalues of GC are $\lambda_i/(\lambda_i + \beta)$, $i = 1, \ldots, s$, and lie in the

Unfortunately, however, while PVI and its scaled version (6.44) are conceptually important, they are not practical algorithms for problems where n is very large. The reason is that the vector $T(\Phi r_k)$ is n-dimensional and its calculation is prohibitive for the type of large problems that we aim to address. Furthermore, even if $T(\Phi r_k)$ were calculated, its projection on S requires knowledge of the steady-state probabilities $\xi_1, \ldots, \xi_n$, which are generally unknown. Fortunately, both of these difficulties can be dealt with through the use of simulation, as we discuss next.

6.3.3 Simulation-Based Methods

We will now consider approximate versions of the methods for solving the projected equation, which involve simulation and low-dimensional calculations. The idea is very simple: we collect simulation samples from the Markov chain associated with the policy, and we average them to form a matrix C_k that approximates

$$C = \Phi' \Xi (I - \alpha P)\Phi,$$

and a vector d_k that approximates

$$d = \Phi' \Xi g;$$

[cf. Eq. (6.41)]. We then approximate the solution $C^{-1}d$ of the projected equation with $C_k^{-1}d_k$, or we approximate the term $(Cr_k - d)$ in the PVI iteration (6.43) [or its scaled version (6.44)] with $(C_k r_k - d_k)$.

The simulation can be done as follows: we generate an infinitely long trajectory $(i_0, i_1, \ldots)$ of the Markov chain, starting from an arbitrary state i_0. After generating state i_t, we compute the corresponding row $\phi(i_t)'$ of Φ, and after generating the transition (i_t, i_{t+1}), we compute the corresponding cost component $g(i_t, i_{t+1})$. After collecting $k+1$ samples $(k = 0, 1, \ldots)$, we form

$$C_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\big(\phi(i_t) - \alpha\phi(i_{t+1})\big)', \tag{6.48}$$

---

interval $(0, 1)$. Actually, the iteration

$$r_{k+1} = r_k - G(Cr_k - d),$$

[cf. Eq. (6.44)], where G is given by Eq. (6.47), is the so-called proximal point algorithm applied to the problem of minimizing $(Cr-d)'\Sigma^{-1}(Cr-d)$ over r. From known results about this algorithm (Martinet [Mar70] and Rockafellar [Roc76]) it follows that the iteration will converge to a minimizing point of $(Cr-d)'\Sigma^{-1}(Cr-d)$. Thus it will converge to some solution of the projected equation $Cr = d$, even if there exist many solutions (as in the case where Φ does not have rank s).

and

$$d_k = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)g(i_t,i_{t+1}), \tag{6.49}$$

where $\phi(i)'$ denotes the ith row of Φ.

It can be proved using simple law of large numbers arguments that $C_k \to C$ and $d_k \to d$ with probability 1. To show this, we use the expression $\Phi' = \big[\phi(1)\cdots\phi(n)\big]$ to write C explicitly as

$$C = \Phi'\Xi(I-\alpha P)\Phi = \sum_{i=1}^{n}\xi_i\phi(i)\left(\phi(i)-\alpha\sum_{j=1}^{n}p_{ij}\phi(j)\right)', \tag{6.50}$$

and we rewrite C_k in a form that matches the preceding expression, except that the probabilities ξ_i and p_{ij} are replaced by corresponding empirical frequencies produced by the simulation. Indeed, by denoting $\delta(\cdot)$ the indicator function [$\delta(E) = 1$ if the event E has occurred and $\delta(E) = 0$ otherwise], we have

$$\begin{aligned} C_k &= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\sum_{t=0}^{k}\delta(i_t=i,i_{t+1}=j)}{k+1}\Big(\phi(i)\big(\phi(i)-\alpha\phi(j)\big)'\Big) \\ &= \sum_{i=1}^{n}\frac{\sum_{t=0}^{k}\delta(i_t=i)}{k+1}\phi(i)\left(\phi(i)-\alpha\sum_{j=1}^{n}\frac{\sum_{t=0}^{k}\delta(i_t=i,i_{t+1}=j)}{\sum_{t=0}^{k}\delta(i_t=i)}\phi(j)\right)' \end{aligned}$$

and finally

$$C_k = \sum_{i=1}^{n}\hat{\xi}_{i,k}\phi(i)\left(\phi(i)-\alpha\sum_{j=1}^{n}\hat{p}_{ij,k}\phi(j)\right)',$$

where

$$\hat{\xi}_{i,k} = \frac{\sum_{t=0}^{k}\delta(i_t=i)}{k+1}, \qquad \hat{p}_{ij,k} = \frac{\sum_{t=0}^{k}\delta(i_t=i,i_{t+1}=j)}{\sum_{t=0}^{k}\delta(i_t=i)}. \tag{6.51}$$

Here, $\hat{\xi}_{i,k}$ and $\hat{p}_{ij,k}$ are the fractions of time that state i, or transition (i,j) has occurred within $(i_0,\ldots,i_k)$, the initial $(k+1)$-state portion of the simulated trajectory. Since the empirical frequencies $\hat{\xi}_{i,k}$ and $\hat{p}_{ij,k}$ asymptotically converge (with probability 1) to the probabilities ξ_i and p_{ij}, respectively, we have with probability 1,

$$C_k \to \sum_{i=1}^{n}\xi_i\phi(i)\left(\phi(i)-\alpha\sum_{j=1}^{n}p_{ij}\phi(j)\right)' = \Phi'\Xi(I-\alpha P)\Phi = C,$$

[cf. Eq. (6.50)]. Similarly, we can write

$$d_k = \sum_{i=1}^{n} \hat{\xi}_{i,k}\phi(i) \sum_{j=1}^{n} \hat{p}_{ij,k} g(i,j),$$

and we have

$$d_k \to \sum_{i=1}^{n} \xi\phi(i) \sum_{j=1}^{n} p_{ij} g(i,j) = \Phi' \Xi g = d.$$

Note that from Eqs. (6.48)-(6.49), C_k and d_k can be updated in a manner reminiscent of stochastic iterative methods, as new samples $\phi(i_k)$ and $g(i_k, i_{k+1})$ are generated. In particular, we have

$$C_k = (1-\delta_k)C_{k-1} + \delta_k \phi(i_k)\big(\phi(i_k) - \alpha\phi(i_{k+1})\big)',$$

$$d_k = (1-\delta_k)d_{k-1} + \delta_k \phi(i_k) g(i_k, i_{k+1}),$$

with the initial conditions $C_{-1} = 0$, $d_{-1} = 0$, and

$$\delta_k = \frac{1}{k+1}, \quad k = 0, 1, \ldots.$$

In these update formulas, δ_k can be viewed as a stepsize, and indeed it can be shown that C_k and d_k converge to C and d for other choices of δ_k (see [Yu10a,b]).

6.3.4 LSTD, LSPE, and TD(0) Methods

Given the simulation-based approximations C_k and d_k of Eqs. (6.48) and (6.49), one possibility is to construct a simulation-based approximate solution

$$\hat{r}_k = C_k^{-1} d_k. \tag{6.52}$$

This is known as the *LSTD (least squares temporal differences)* method. Despite the dependence on the index k, this is not an iterative method, since $\hat{r}_{k-1}$ is not needed to compute $\hat{r}_k$. Rather it may be viewed as an *approximate matrix inversion approach*: we replace the projected equation $Cr = d$ with the approximation $C_k r = d_k$, using a batch of $k+1$ simulation samples, and solve the approximate equation by matrix inversion. Note that by using Eqs. (6.48) and (6.49), the equation $C_k r = d_k$ can be written as

$$C_k r - d_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) q_{k,t} = 0, \tag{6.53}$$

where

$$q_{k,t} = \phi(i_t)' r_k - \alpha\phi(i_{t+1})' r_k - g(i_t, i_{t+1}). \tag{6.54}$$

The scalar $q_{k,t}$ is the so-called *temporal difference*, associated with r_k and transition (i_t, i_{t+1}). It may be viewed as a sample of a residual term arising in the projected Bellman's equation. More specifically, from Eqs. (6.40), (6.41), we have

$$Cr_k - d = \Phi'\Xi(\Phi r_k - \alpha P \Phi r_k - g). \tag{6.55}$$

The three terms in the definition (6.54) of the temporal difference $q_{k,t}$ can be viewed as samples [associated with the transition (i_t, i_{t+1})] of the corresponding three terms in the expression $\Xi(\Phi r_k - \alpha P \Phi r_k - g)$ in Eq. (6.55).

Regression-Based LSTD

An important concern in LSTD is to ensure that the simulation-induced error

$$e_k = \Phi(\hat{r}_k - r^*) = \Phi(C_k^{-1} d_k - C^{-1} d)$$

is not excessively large. Then the low-dimensional error $C_k^{-1} d_k - C^{-1} d$ is typically also large (the reverse is not true: $\hat{r}_k - r^*$ may be large without e_k being large). In the lookup table representation case ($\Phi = I$) a large error e_k may be traced directly to the simulation error in evaluating C and d, combined with near singularity of $\Xi(I - \alpha P)$. In the compact representation case ($\Phi \neq I$), the effect of near singularity of C on the high-dimensional error e_k is more complex, but is also primarily due to the same causes.† In what follows we will consider approaches to reduce the low-dimensional error $\hat{r}_k - r^*$ with the understanding that these approaches will also be effective in reducing the high-dimensional error e_k, when the latter is very large.

† Near-singularity of C, causing large low-dimensional errors $C_k^{-1} d_k - C^{-1} d$, may be due either to the columns of Φ being nearly linearly dependent or to the matrix $\Xi(I - \alpha P)$ being nearly singular [cf. the formula $C = \Phi'\Xi(I - \alpha P)\Phi$ of Eq. (6.41)]. However, near-linear dependence of the columns of Φ will not affect the high-dimensional error e_k. The reason is that e_k depends only on the subspace S and not its representation in terms of the matrix Φ. In particular, if we replace Φ with a matrix ΦB where B is an $s \times s$ invertible scaling matrix, the subspace S will be unaffected and the errors e_k will also be unaffected, as can be verified using the formulas of Section 6.3.3. On the other hand, near singularity of the matrix $I - \alpha P$ may affect significantly e_k. Note that $I - \alpha P$ is nearly singular in the case where α is very close to 1, or in the corresponding undiscounted case where $\alpha = 1$ and P is substochastic with eigenvalues very close to 1 (see Section 6.6). Large variations in the size of the diagonal components of Ξ may also affect significantly e_k, although this dependence is complicated by the fact that Ξ appears not only in the formula $C = \Phi'\Xi(I - \alpha P)\Phi$ but also in the formula $d = \Phi'\Xi g$.

Example 6.3.1

To get a rough sense of the potential effect of the simulation error in LSTD, consider the approximate inversion of a small nonzero number c, which is estimated with simulation error ϵ. The absolute and relative errors are

$$E = \frac{1}{c+\epsilon} - \frac{1}{c}, \qquad E_r = \frac{E}{1/c}.$$

By a first order Taylor series expansion around $\epsilon = 0$, we obtain for small ϵ

$$E \approx \frac{\partial\big(1/(c+\epsilon)\big)}{\partial\epsilon}\Big|_{\epsilon=0} \epsilon = -\frac{\epsilon}{c^2}, \qquad E_r \approx -\frac{\epsilon}{c}.$$

Thus for the estimate $\frac{1}{c+\epsilon}$ to be reliable, we must have $|\epsilon| << |c|$. If N independent samples are used to estimate c, the variance of ϵ is proportional to $1/N$, so for a small relative error, N must be much larger than $1/c^2$. Thus as c approaches 0, the amount of sampling required for reliable simulation-based inversion increases very fast.

To reduce the size of the errors $\hat{r}_k - r^*$, an effective remedy is to estimate r^* by a form of regularized regression, which works even if C_k is singular, at the expense of a systematic/deterministic error (a "bias") in the generated estimate. In this approach, instead of solving the system $C_k r = d_k$, we use a least-squares fit of a linear model that properly encodes the effect of the simulation noise.

We write the projected form of Bellman's equation $d = Cr$ as

$$d_k = C_k r + e_k, \tag{6.56}$$

where e_k is the vector

$$e_k = (C - C_k)r + d_k - d,$$

which we view as "simulation noise." We then estimate the solution r^* based on Eq. (6.56) by using regression. In particular, we choose r by solving the least squares problem:

$$\min_r \big\{(d_k - C_k r)'\Sigma^{-1}(d_k - C_k r) + \beta\|r - \bar{r}\|^2\big\}, \tag{6.57}$$

where $\bar{r}$ is an *a priori* estimate of r^*, Σ is some positive definite symmetric matrix, and β is a positive scalar. By setting to 0 the gradient of the least squares objective in Eq. (6.57), we can find the solution in closed form:

$$\hat{r}_k = (C_k'\Sigma^{-1}C_k + \beta I)^{-1}(C_k'\Sigma^{-1}d_k + \beta\bar{r}). \tag{6.58}$$

A suitable choice of $\bar{r}$ may be some heuristic guess based on intuition about the problem, or it may be the parameter vector corresponding to the estimated cost vector $\Phi\bar{r}$ of a similar policy (for example a preceding policy

in an approximate policy iteration context). One may try to choose Σ in special ways to enhance the quality of the estimate of r^*, but we will not consider this issue here, and the subsequent analysis in this section does not depend on the choice of Σ, as long as it is positive definite and symmetric.

The quadratic $\beta\|r-\bar{r}\|^2$ in Eq. (6.57) is known as a *regularization term*, and has the effect of "biasing" the estimate $\hat{r}_k$ towards the *a priori* guess $\bar{r}$. The proper size of β is not clear (a large size reduces the effect of near singularity of C_k, and the effect of the simulation errors $C_k - C$ and $d_k - d$, but may also cause a large "bias"). However, this is typically not a major difficulty in practice, because trial-and-error experimentation with different values of β involves low-dimensional linear algebra calculations once C_k and d_k become available.

We will now derive an estimate for the error $\hat{r}_k - r^*$, where $r^* = C^{-1}d$ is the solution of the projected equation. Let us denote

$$b_k = \Sigma^{-1/2}(d_k - C_k r^*),$$

so from Eq. (6.58),

$$\hat{r}_k - r^* = (C_k'\Sigma^{-1}C_k + \beta I)^{-1}\left(C_k'\Sigma^{-1/2}b_k + \beta(\bar{r}-r^*)\right). \tag{6.59}$$

We have the following proposition, which involves the singular values of the matrix $\Sigma^{-1/2}C_k$ (these are the square roots of the eigenvalues of $C_k'\Sigma^{-1}C_k$; see e.g., [Str09], [TrB97]).

Proposition 6.3.4: We have

$$\|\hat{r}_k - r^*\| \le \max_{i=1,\ldots,s}\left\{\frac{\lambda_i}{\lambda_i^2+\beta}\right\}\|b_k\| + \max_{i=1,\ldots,s}\left\{\frac{\beta}{\lambda_i^2+\beta}\right\}\|\bar{r}-r^*\|, \tag{6.60}$$

where $\lambda_1,\ldots,\lambda_s$ are the singular values of $\Sigma^{-1/2}C_k$.

Proof: Let $\Sigma^{-1/2}C_k = U\Lambda V'$ be the singular value decomposition of $\Sigma^{-1/2}C_k$, where $\Lambda = \mathrm{diag}\{\lambda_1,\ldots,\lambda_s\}$, and U, V are unitary matrices ($UU' = VV' = I$ and $\|U\| = \|U'\| = \|V\| = \|V'\| = 1$; see [Str09], [TrB97]). Then, Eq. (6.59) yields

$$\begin{aligned}\hat{r}_k - r^* &= (V\Lambda U'U\Lambda V' + \beta I)^{-1}\left(V\Lambda U'b_k + \beta(\bar{r}-r^*)\right)\\ &= (V')^{-1}(\Lambda^2+\beta I)^{-1}V^{-1}\left(V\Lambda U'b_k + \beta(\bar{r}-r^*)\right)\\ &= V(\Lambda^2+\beta I)^{-1}\Lambda U'b_k + \beta\, V(\Lambda^2+\beta I)^{-1}V'(\bar{r}-r^*).\end{aligned}$$

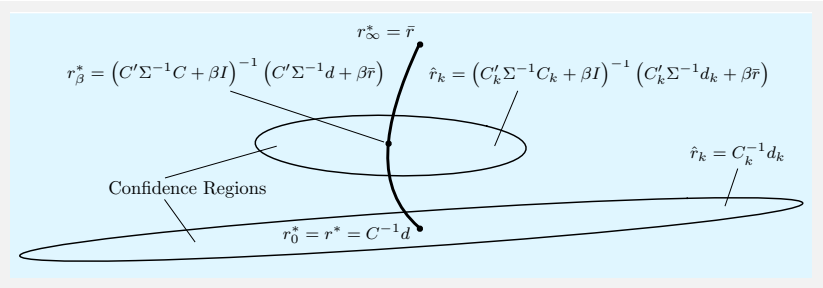

Figure 6.3.3 Illustration of Prop. 6.3.4. The figure shows the estimates

$$\hat{r}_k = \left(C_k'\Sigma^{-1}C_k + \beta I\right)^{-1}\left(C_k'\Sigma^{-1}d_k + \beta\bar{r}\right)$$

corresponding to a finite number of samples, and the exact values

$$r^*_\beta = \left(C'\Sigma^{-1}C + \beta I\right)^{-1}\left(C'\Sigma^{-1}d + \beta\bar{r}\right)$$

corresponding to an infinite number of samples. We may view $\hat{r}_k - r^*$ as the sum of a "simulation error" $\hat{r}_k - r^*_\beta$ whose norm is bounded by the first term in the estimate (6.60) and can be made arbitrarily small by sufficiently long sampling, and a "regularization error" $r^*_\beta - r^*$ whose norm is bounded by the second term in the right-hand side of Eq. (6.60).

Therefore, using the triangle inequality, we have

$$\begin{aligned}\|\hat{r}_k - r^*\| &\le \|V\| \max_{i=1,\ldots,s}\left\{\frac{\lambda_i}{\lambda_i^2+\beta}\right\}\|U'\|\,\|b_k\| \\ &\quad + \beta\|V\| \max_{i=1,\ldots,s}\left\{\frac{1}{\lambda_i^2+\beta}\right\}\|V'\|\,\|\bar{r} - r^*\| \\ &= \max_{i=1,\ldots,s}\left\{\frac{\lambda_i}{\lambda_i^2+\beta}\right\}\|b_k\| + \max_{i=1,\ldots,s}\left\{\frac{\beta}{\lambda_i^2+\beta}\right\}\|\bar{r} - r^*\|.\end{aligned}$$

Q.E.D.

From Eq. (6.60), we see that the error $\|\hat{r}_k - r^*\|$ is bounded by the sum of two terms. The first term can be made arbitrarily small by using a sufficiently large number of samples, thereby making $\|b_k\|$ small. The second term reflects the bias introduced by the regularization and diminishes with β, but it cannot be made arbitrarily small by using more samples (see Fig. 6.3.3).

Now consider the case where $\beta = 0$, Σ is the identity, and C_k is invertible. Then $\hat{r}_k$ is the LSTD solution $C_k^{-1}d_k$, and the proof of Prop. 6.3.4 can be replicated to show that

$$\|\hat{r}_k - r^*\| \le \max_{i=1,\ldots,s}\left\{\frac{1}{\lambda_i}\right\}\|b_k\|,$$

where $\lambda_1, \ldots, \lambda_s$ are the (positive) singular values of C_k. This suggests that without regularization, the LSTD error can be adversely affected by near singularity of the matrix C_k (smallest λ_i close to 0). Thus we expect that for a nearly singular matrix C, a very large number of samples are necessary to attain a small error $(\hat{r}_k - r^*)$, with serious difficulties potentially resulting, consistent with the scalar inversion example we gave earlier.

We also note an alternative and somewhat simpler regularization approach, whereby we approximate the equation $C_k r = d_k$ by

$$(C_k + \beta I)\hat{r} = d_k + \beta \bar{r}, \tag{6.61}$$

where β is a positive scalar and $\bar{r}$ is some guess of the solution $r^* = C^{-1}d$. We refer to [Ber11a] for more details on this method.

Generally, the regularization of LSTD alleviates the effects of near singularity of C and simulation error, but it comes at a price: there is a bias of the estimate $\hat{r}_k$ towards the prior guess $\bar{r}$ (cf. Fig. 6.3.3). One possibility to eliminate this bias is to adopt an iterative regularization approach: start with some $\bar{r}$, obtain $\hat{r}_k$, replace $\bar{r}$ by $\hat{r}_k$, and repeat for any number of times. This turns the regression-based LSTD method (6.58) to the iterative method

$$\hat{r}_{k+1} = (C_k' \Sigma_k^{-1} C_k + \beta I)^{-1}(C_k' \Sigma_k^{-1} d_k + \beta \hat{r}_k), \tag{6.62}$$

which will be shown to be a special case of the class of iterative LSPE-type methods to be discussed later.

LSPE Method

We will now develop a simulation-based implementation of the PVI iteration

$$\Phi r_{k+1} = \Pi T(\Phi r_k).$$

By expressing the projection as a least squares minimization, we see that r_{k+1} is given by

$$r_{k+1} = \arg\min_{r \in \Re^s} \big\|\Phi r - T(\Phi r_k)\big\|_\xi^2,$$

or equivalently

$$r_{k+1} = \arg\min_{r \in \Re^s} \sum_{i=1}^n \xi_i \left(\phi(i)'r - \sum_{j=1}^n p_{ij}\big(g(i,j) + \alpha \phi(j)' r_k\big) \right)^2. \tag{6.63}$$

We approximate this optimization by generating an infinitely long trajectory $(i_0, i_1, \ldots)$ and by updating r_k after each transition (i_k, i_{k+1}) according to

$$r_{k+1} = \arg\min_{r \in \Re^s} \sum_{t=0}^k \big(\phi(i_t)'r - g(i_t, i_{t+1}) - \alpha \phi(i_{t+1})' r_k\big)^2. \tag{6.64}$$

We call this iteration *least squares policy evaluation* (LSPE for short).

The similarity of PVI [Eq. (6.63)] and LSPE [Eq. (6.64)] can be seen by explicitly calculating the solutions of the associated least squares problems. For PVI, by setting the gradient of the cost function in Eq. (6.63) to 0 and using a straightforward calculation, we have

$$r_{k+1} = \left(\sum_{i=1}^{n} \xi_i\, \phi(i)\phi(i)'\right)^{-1} \left(\sum_{i=1}^{n} \xi_i\, \phi(i) \sum_{j=1}^{n} p_{ij}\big(g(i,j) + \alpha\phi(j)'r_k\big)\right). \tag{6.65}$$

For LSPE, we similarly have from Eq. (6.64)

$$r_{k+1} = \left(\sum_{t=0}^{k} \phi(i_t)\phi(i_t)'\right)^{-1} \left(\sum_{t=0}^{k} \phi(i_t)\big(g(i_t,i_{t+1}) + \alpha\phi(i_{t+1})'r_k\big)\right). \tag{6.66}$$

This equation can equivalently be written as

$$r_{k+1} = \left(\sum_{i=1}^{n} \hat{\xi}_{i,k}\, \phi(i)\phi(i)'\right)^{-1} \left(\sum_{i=1}^{n} \hat{\xi}_{i,k}\, \phi(i) \sum_{j=1}^{n} \hat{p}_{ij,k}\big(g(i,j) + \alpha\phi(j)'r_k\big)\right), \tag{6.67}$$

where $\hat{\xi}_{i,k}$ and $\hat{p}_{ij,k}$ are empirical frequencies of state i and transition (i,j), defined by

$$\hat{\xi}_{i,k} = \frac{\sum_{t=0}^{k} \delta(i_t = i)}{k+1}, \qquad \hat{p}_{ij,k} = \frac{\sum_{t=0}^{k} \delta(i_t = i, i_{t+1} = j)}{\sum_{t=0}^{k} \delta(i_t = i)}. \tag{6.68}$$

(We will discuss later the question of existence of the matrix inverses in the preceding equations.) Here, $\delta(\cdot)$ denotes the indicator function [$\delta(E) = 1$ if the event E has occurred and $\delta(E) = 0$ otherwise], so for example, $\hat{\xi}_{i,k}$ is the fraction of time that state i has occurred within $(i_0, \ldots, i_k)$, the initial $(k+1)$-state portion of the simulated trajectory. By comparing Eqs. (6.65) and (6.67), we see that they asymptotically coincide, since the empirical frequencies $\hat{\xi}_{i,k}$ and $\hat{p}_{ij,k}$ asymptotically converge (with probability 1) to the probabilities ξ_i and p_{ij}, respectively.

Thus, LSPE may be viewed as PVI with simulation error added in the right-hand side (see Fig. 6.3.3). Since the empirical frequencies $\hat{\xi}_{i,k}$ and $\hat{p}_{ij,k}$ converge to the probabilities ξ_i and p_{ij}, the error asymptotically diminishes to 0 (assuming the iterates produced by LSPE are bounded). Because of this diminishing nature of the error and the contraction property of ΠT, it is intuitively clear and can be rigorously shown that LSPE converges to the same limit as PVI. The limit is the unique r^* satisfying the equation

$$\Phi r^* = \Pi T(\Phi r^*)$$

[cf. Eq. (6.37)], and the error estimate of Prop. 6.3.2 applies. LSPE may also be viewed as a special case of the class of simulation-based versions of the deterministic iterative methods of Section 6.3.2, which we discuss next.

Other Iterative Simulation-Based Methods

An alternative to LSTD is to use a true iterative method to solve the projected equation $Cr = d$ using simulation-based approximations to C and d. One possibility is to approximate the scaled PVI iteration [cf. Eq. (6.44)]

$$r_{k+1} = r_k - \gamma G(Cr_k - d) \tag{6.69}$$

with

$$r_{k+1} = r_k - \gamma \hat{G}(\hat{C}r_k - \hat{d}), \tag{6.70}$$

where $\hat{C}$ and $\hat{d}$ are simulation-based estimates of C and d, γ is a positive stepsize, and $\hat{G}$ is an $s \times s$ matrix, which may also be obtained by simulation. Assuming that $I - \gamma\hat{G}\hat{C}$ is a contraction, this iteration will yield a solution to the system $\hat{C}r = \hat{d}$, which will serve as a simulation-based approximation to a solution of the projected equation $Cr = d$.

Like LSTD, this may be viewed as a batch simulation approach: we first simulate to obtain $\hat{C}$, $\hat{d}$, and $\hat{G}$, and then solve the system $\hat{C}r = \hat{d}$ by the iteration (6.70) rather than direct matrix inversion. An alternative is to iteratively update r as simulation samples are collected and used to form ever improving approximations to C and d. In particular, one or more iterations of the form (6.70) may be performed after collecting a few additional simulation samples that are used to improve the approximations of the current $\hat{C}$ and $\hat{d}$. In the most extreme type of such an algorithm, the iteration (6.70) is used after a single new sample is collected. This algorithm has the form

$$r_{k+1} = r_k - \gamma G_k(C_k r_k - d_k), \tag{6.71}$$

where G_k is an $s \times s$ matrix, γ is a positive stepsize, and C_k and d_k are given by Eqs. (6.48)-(6.49). For the purposes of further discussion, we will focus on this algorithm, with the understanding that there are related versions that use (partial) batch simulation and have similar properties. Note that the iteration (6.71) may also be written in terms of temporal differences as

$$r_{k+1} = r_k - \frac{\gamma}{k+1} G_k \sum_{t=0}^{k} \phi(i_t) q_{k,t} \tag{6.72}$$

[cf. Eqs. (6.48), (6.49), (6.54)]. The convergence behavior of this method is satisfactory. Generally, we have $r_k \to r^*$, provided $C_k \to C$, $d_k \to d$, and $G_k \to G$, where G and γ are such that $I - \gamma GC$ is a contraction [this is fairly evident in view of the convergence of the iteration (6.69), which was shown in Section 6.3.2; see also the papers [Ber09b], [Ber11a]].

To ensure that $I - \gamma GC$ is a contraction for small γ, we may choose G to be symmetric and positive definite, or to have a special form, such as

$$G = (C'\Sigma^{-1}C + \beta I)^{-1}C'\Sigma^{-1},$$

where Σ is any positive definite symmetric matrix, and β is a positive scalar [cf. Eq. (6.47)].

Regarding the choices of γ and G_k, one possibility is to choose $\gamma = 1$ and G_k to be a simulation-based approximation to $G = (\Phi'\Xi\Phi)^{-1}$, which is used in the PVI method (6.42)-(6.43):

$$G_k = \left(\frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\phi(i_t)' \right)^{-1}, \tag{6.73}$$

or

$$G_k = \left(\frac{\beta}{k+1} I + \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\phi(i_t)' \right)^{-1}, \tag{6.74}$$

where βI is a positive multiple of the identity (to ensure that G_k is positive definite). Note that when $\gamma = 1$ and G_k is given by Eq. (6.73), the iteration (6.71) is identical to the LSPE iteration (6.66) [cf. the forms of C_k and d_k given by Eqs. (6.48) and (6.49)].

While G_k, as defined by Eqs. (6.73) and (6.74), requires updating and inversion at every iteration, a partial batch mode of updating G_k is also possible: one may introduce into iteration (6.71) a new estimate of $G = (\Phi'\Xi\Phi)^{-1}$ periodically, obtained from the previous estimate using multiple simulation samples. This will save some computation and will not affect the asymptotic convergence rate of the method, as we will discuss shortly. Indeed, as noted earlier, the iteration (6.71) itself may be executed in partial batch mode, after collecting multiple samples between iterations. Note also that even if G_k is updated at every k using Eqs. (6.73) and (6.74), the updating can be done recursively; for example, from Eq. (6.73) we have

$$G_k^{-1} = \frac{k}{k+1} G_{k-1}^{-1} + \frac{1}{k+1} \phi(i_k)\phi(i_k)'.$$

A simple possibility is to use a diagonal matrix G_k, thereby simplifying the matrix inversion in the iteration (6.71). One possible choice is a diagonal approximation to $\Phi'\Xi\Phi$, obtained by discarding the off-diagonal terms of the matrix (6.73) or (6.74). Then it is reasonable to expect that a stepsize γ close to 1 will often lead to $I - \gamma GC$ being a contraction, thereby facilitating the choice of γ. The simplest possibility is to just choose G_k to be the identity, although in this case, some experimentation is needed to find a proper value of γ such that $I - \gamma C$ is a contraction.

Another choice of G_k is

$$G_k = (C_k'\Sigma_k^{-1}C_k + \beta I)^{-1} C_k'\Sigma_k^{-1}, \tag{6.75}$$

where Σ_k is some positive definite symmetric matrix, and β is a positive scalar. Then the iteration (6.71) takes the form

$$r_{k+1} = r_k - \gamma (C_k'\Sigma_k^{-1}C_k + \beta I)^{-1} C_k'\Sigma_k^{-1}(C_k r_k - d_k),$$

and for $\gamma = 1$, it can be written as

$$r_{k+1} = (C_k' \Sigma_k^{-1} C_k + \beta I)^{-1} (C_k' \Sigma_k^{-1} d_k + \beta r_k). \tag{6.76}$$

We recognize this as an iterative version of the regression-based LSTD method (6.58), where the prior guess $\bar{r}$ is replaced by the previous iterate r_k [cf. Eq. (6.62)]. This iteration is convergent to r^* provided that $\{\Sigma_k^{-1}\}$ is bounded [$\gamma = 1$ is within the range of stepsizes for which $I - \gamma GC$ is a contraction; see the discussion following Eq. (6.47)].

A simpler regularization-based choice of G_k is

$$G_k = (C_k + \beta I)^{-1}$$

[cf. Eq. (6.61)]. Then the iteration (6.71) takes the form

$$r_{k+1} = r_k - (C_k + \beta I)^{-1} (C_k r_k - d_k). \tag{6.77}$$

The convergence of this iteration can be proved, thanks to the positive definiteness of C [cf. Prop. 6.3.3], based on the fact $C_k \to C$ and standard convergence results for the proximal point algorithm ([Mar70], [Roc76]); see also [Ber11a]. Note that by contrast with Eq. (6.76), the positive definiteness of C is essential both for invertibility of $C_k + \beta I$ and for convergence of Eq. (6.77).

Convergence Rate of Iterative Methods – Comparison with LSTD

Let us now discuss the choice of γ and G from the convergence rate point of view. It can be easily verified with simple examples that the values of γ and G affect significantly the convergence rate of the deterministic scaled PVI iteration (6.69). Surprisingly, however, the asymptotic convergence rate of the simulation-based iteration (6.71) does not depend on the choices of γ and G. Indeed it can be proved that *the iteration (6.71) converges at the same rate asymptotically, regardless of the choices of γ and G, as long as $I - \gamma GC$ is a contraction* (although the short-term convergence rate may be significantly affected by the choices of γ and G).

The reason is that the scaled PVI iteration (6.69) has a linear convergence rate (since it involves a contraction), which is fast relative to the slow convergence rate of the simulation-generated G_k, C_k, and d_k. Thus the simulation-based iteration (6.71) operates on two time scales (see, e.g., Borkar [Bor08], Ch. 6): the slow time scale at which G_k, C_k, and d_k change, and the fast time scale at which r_k adapts to changes in G_k, C_k, and d_k. As a result, essentially, there is convergence in the fast time scale before there is appreciable change in the slow time scale. Roughly speaking, r_k "sees G_k, C_k, and d_k as effectively constant," so that for large k, r_k is essentially equal to the corresponding limit of iteration (6.71) with G_k, C_k, and d_k

held fixed. This limit is $C_k^{-1}d_k$. It follows that the sequence r_k generated by the scaled LSPE iteration (6.71) "tracks" the sequence $C_k^{-1}d_k$ generated by the LSTD iteration in the sense that

$$\|r_k - C_k^{-1}d_k\| << \|r_k - r^*\|, \qquad \text{for large } k,$$

independent of the choice of γ and the scaling matrix G that is approximated by G_k (see also [YuB06b], [Ber09b], [Ber11a] for analysis and further discussion).

TD(0) Method

This is an iterative method for solving the projected equation $Cr = d$. Like LSTD and LSPE, it generates an infinitely long trajectory $\{i_0, i_1, \ldots\}$ of the Markov chain, but at each iteration, it uses only one sample, the last one. It has the form

$$r_{k+1} = r_k - \gamma_k \phi(i_k) q_{k,k}, \tag{6.78}$$

where γ_k is a stepsize sequence that diminishes to 0. It may be viewed as an instance of a classical stochastic approximation scheme for solving the projected equation $Cr = d$. This equation can be written as $\Phi'\Xi(\Phi r - A\Phi r - b) = 0$, and by using Eqs. (6.54) and (6.78), it can be seen that the direction of change $\phi(i_k)q_{k,k}$ in TD(0) is a sample of the left-hand side $\Phi'\Xi(\Phi r - A\Phi r - b)$ of the equation.

Let us note a similarity between TD(0) and the scaled LSPE method (6.72) with $G_k = I$, given by:

$$r_{k+1} = r_k - \gamma(C_k r_k - d_k) = r_k - \frac{\gamma}{k+1}\sum_{t=0}^{k} \phi(i_t) q_{k,t}. \tag{6.79}$$

While LSPE uses as direction of change a time-average approximation of $Cr_k - d$ based on all the available samples, TD(0) *uses a single sample approximation*. It is thus not surprising that TD(0) is a much slower algorithm than LSPE, and moreover requires that the stepsize γ_k diminishes to 0 in order to deal with the nondiminishing noise that is inherent in the term $\phi(i_k)q_{k,k}$ of Eq. (6.78). On the other hand, TD(0) requires much less overhead per iteration: calculating the single temporal difference $q_{k,k}$ and multiplying it with $\phi(i_k)$, rather than updating the $s \times s$ matrix C_k and multiplying it with r_k. Thus when s, the number of features, is very large, TD(0) may offer a significant overhead advantage over LSTD and LSPE.

We finally note a scaled version of TD(0) given by

$$r_{k+1} = r_k - \gamma_k G_k \phi(i_k) q_{k,k}, \tag{6.80}$$

where G_k is a positive definite symmetric scaling matrix, selected to speed up convergence. It is a scaled (by the matrix G_k) version of TD(0), so it may be viewed as a type of scaled stochastic approximation method.

6.3.5 Optimistic Versions

In the LSTD and LSPE methods discussed so far, the underlying assumption is that each policy is evaluated with a very large number of samples, so that an accurate approximation of C and d are obtained. There are also optimistic versions (cf. Section 6.1.2), where the policy μ is replaced by an "improved" policy $\overline{\mu}$ after only a certain number of simulation samples have been processed.

A natural form of optimistic LSTD is $\hat{r}_{k+1} = C_k^{-1} d_k$, where C_k and d_k are obtained by averaging samples collected using the controls corresponding to the (approximately) improved policy. By this we mean that C_k and d_k are time averages of the matrices and vectors

$$\phi(i_t)\big(\phi(i_t) - \alpha\phi(i_{t+1})\big)', \qquad \phi(i_t)g(i_t, i_{t+1}),$$

corresponding to simulated transitions (i_t, i_{t+1}) that are generated using the policy μ^{k+1} whose controls are given by

$$\mu^{k+1}(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha\phi(j)'\hat{r}_k\big)$$

[cf. Eq. (6.52)]. Unfortunately, this method requires the collection of many samples between policy updates, as it is susceptible to simulation noise in C_k and d_k.

The optimistic version of (scaled) LSPE is based on similar ideas. Following the state transition (i_k, i_{k+1}), we update r_k using the iteration

$$r_{k+1} = r_k - \gamma G_k(C_k r_k - d_k), \tag{6.81}$$

where C_k and d_k are given by Eqs. (6.48), (6.49) [cf. Eq. (6.71)], and G_k is a scaling matrix that converges to some G for which $I - \gamma GC$ is a contraction. For example G_k could be a positive definite symmetric matrix [such as for example the one given by Eq. (6.73)] or the matrix

$$G_k = (C_k'\Sigma_k^{-1}C_k + \beta I)^{-1}C_k'\Sigma_k^{-1} \tag{6.82}$$

[cf. Eq. (6.75)]. In the latter case, for $\gamma = 1$ the method takes the form

$$\hat{r}_{k+1} = (C_k'\Sigma_k^{-1}C_k + \beta I)^{-1}(C_k'\Sigma_k^{-1}d_k + \beta\hat{r}_k), \tag{6.83}$$

[cf. Eq. (6.76)]. The simulated transitions are generated using a policy that is updated every few samples. In the extreme case of a single sample between policies, we generate the next transition (i_{k+1}, i_{k+2}) using the control

$$u_{k+1} = \arg\min_{u \in U(i_{k+1})} \sum_{j=1}^{n} p_{i_{k+1}j}(u)\big(g(i_{k+1}, u, j) + \alpha\phi(j)'r_{k+1}\big).$$

Because the theoretical convergence guarantees of LSPE apply only to the nonoptimistic version, it may be essential to experiment with various values of the stepsize γ [this is true even if G_k is chosen according to Eq. (6.73), for which $\gamma = 1$ guarantees convergence in the nonoptimistic version]. There is also a similar optimistic version of TD(0).

To improve the reliability of the optimistic LSTD method it seems necessary to turn it into an iterative method, which then brings it closer to LSPE. In particular, an iterative version of the regression-based LSTD method (6.58) is given by Eq. (6.83), and is the special case of LSPE, corresponding to the special choice of the scaling matrix G_k of Eq. (6.82).

Generally, in optimistic LSTD and LSPE, a substantial number of samples may need to be collected with the same policy before switching policies, in order to reduce the variance of C_k and d_k. As an alternative, one may consider building up C_k and d_k as weighted averages, using samples from several past policies, while giving larger weight to the samples of the current policy. One may argue that mixing samples from several past policies may have a beneficial exploration effect. Still, however, similar to other versions of policy iteration, to enhance exploration, one may occasionally introduce randomly transitions other than the ones dictated by the current policy (cf. the discussion of Section 6.1.2). The complexities introduced by these variations are not fully understood at present. For experimental investigations of optimistic policy iteration, see Bertsekas and Ioffe [BeI96], Jung and Polani [JuP07], Busoniu et al. [BED09], and Thiery and Scherrer [ThS10a].

6.3.6 Multistep Simulation-Based Methods

A useful approach in approximate DP is to replace Bellman's equation with an equivalent equation that reflects control over multiple successive stages. This amounts to replacing T with a multistep version that has the same fixed points; for example, T^ℓ with $\ell > 1$, or $T^{(\lambda)}$ given by

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

where $\lambda \in (0,1)$. We will focus on the λ-weighted multistep Bellman equation

$$J = T^{(\lambda)} J.$$

By noting that

$$T^2 J = g + \alpha P(TJ) = g + \alpha P(g + \alpha PJ) = (I + \alpha P)g + \alpha^2 P^2 J,$$

$$T^3 J = g + \alpha P(T^2 J) = g + \alpha P\big((I+\alpha P)g + \alpha^2 P^2 J\big) = (I + \alpha P + \alpha^2 P^2)g + \alpha^3 P^3 J,$$

etc, this equation can be written as

$$J = T^{(\lambda)} J = g^{(\lambda)} + \alpha P^{(\lambda)} J, \tag{6.84}$$

with

$$P^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty}\alpha^{\ell}\lambda^{\ell}P^{\ell+1}, \qquad g^{(\lambda)} = \sum_{\ell=0}^{\infty}\alpha^{\ell}\lambda^{\ell}P^{\ell}g = (I-\alpha\lambda P)^{-1}g. \tag{6.85}$$

We may then apply variants of the preceding simulation algorithms to find a fixed point of $T^{(\lambda)}$ in place of T. The corresponding projected equation takes the form

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

where

$$C^{(\lambda)} = \Phi'\Xi\big(I-\alpha P^{(\lambda)}\big)\Phi, \qquad d^{(\lambda)} = \Phi'\Xi g^{(\lambda)}, \tag{6.86}$$

[cf. Eq. (6.41)]. The motivation for replacing T with $T^{(\lambda)}$ is that the modulus of contraction of $T^{(\lambda)}$ is smaller, resulting in a tighter error bound. This is shown in the following proposition.

Proposition 6.3.5: The mappings $T^{(\lambda)}$ and $\Pi T^{(\lambda)}$ are contractions of modulus

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

with respect to the weighted Euclidean norm $\|\cdot\|_\xi$, where ξ is the steady-state probability vector of the Markov chain. Furthermore

$$\|J_\mu - \Phi r_\lambda^*\|_\xi \le \frac{1}{\sqrt{1-\alpha_\lambda^2}}\,\|J_\mu - \Pi J_\mu\|_\xi, \tag{6.87}$$

where Φr_λ^* is the fixed point of $\Pi T^{(\lambda)}$.

Proof: Using Lemma 6.3.1, we have

$$\begin{aligned}\|P^{(\lambda)}z\|_\xi &\le (1-\lambda)\sum_{\ell=0}^{\infty}\alpha^{\ell}\lambda^{\ell}\|P^{\ell+1}z\|_\xi \\ &\le (1-\lambda)\sum_{\ell=0}^{\infty}\alpha^{\ell}\lambda^{\ell}\|z\|_\xi \\ &= \frac{(1-\lambda)}{1-\alpha\lambda}\|z\|_\xi.\end{aligned}$$

Since $T^{(\lambda)}$ is linear with associated matrix $\alpha P^{(\lambda)}$ [cf. Eq. (6.84)], it follows that $T^{(\lambda)}$ is a contraction with modulus $\alpha(1-\lambda)/(1-\alpha\lambda)$. The estimate (6.87) follows similar to the proof of Prop. 6.3.2. **Q.E.D.**

Note that α_λ decreases as λ increases, and $\alpha_\lambda \to 0$ as $\lambda \to 1$. Furthermore, the error bound (6.87) becomes better as λ increases. Indeed

from Eq. (6.87), it follows that as $\lambda \to 1$, the projected equation solution Φr_λ^* converges to the "best" approximation ΠJ_μ of J_μ on S. This suggests that large values of λ should be used. On the other hand, we will later argue that when simulation-based approximations are used, the effects of simulation noise become more pronounced as λ increases. Furthermore, we should note that in the context of approximate policy iteration, the objective is not just to approximate well the cost of the current policy, but rather to use the approximate cost to obtain the next "improved" policy. We are ultimately interested in a "good" next policy, and there is no consistent experimental or theoretical evidence that this is achieved solely by good cost approximation of the current policy. Thus, in practice, some trial and error with the value of λ may be useful.

Another interesting fact, which follows from $\lim_{\lambda \to 1} \alpha_\lambda = 0$, is that given any norm, the mapping $T^{(\lambda)}$ is a contraction (with arbitrarily small modulus) with respect to that norm for λ sufficiently close to 1. This is a consequence of the norm equivalence property in $\Re^n$ (any norm is bounded by a constant multiple of any other norm). As a result, for any weighted Euclidean norm of projection, $\Pi T^{(\lambda)}$ is a contraction provided λ is sufficiently close to 1.

LSTD(λ), LSPE(λ), and TD(λ)

The simulation-based methods of the preceding subsections correspond to $\lambda = 0$, but can be extended to $\lambda > 0$. In particular, in a matrix inversion approach, the unique solution of the projected equation may be approximated by

$$\big(C_k^{(\lambda)}\big)^{-1} d_k^{(\lambda)}, \tag{6.88}$$

where $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ are simulation-based approximations of $C^{(\lambda)}$ and $d^{(\lambda)}$, given by Eq. (6.86). This is the LSTD(λ) method. There is also a regression/regularization variant of this method along the lines described earlier [cf. Eq. (6.58)].

Similarly, we may consider the (scaled) LSPE(λ) iteration

$$r_{k+1} = r_k - \gamma G_k \big(C_k^{(\lambda)} r_k - d_k^{(\lambda)}\big), \tag{6.89}$$

where γ is a stepsize and G_k is a scaling matrix that converges to some G such that $I - \gamma G C^{(\lambda)}$ is a contraction. One possibility is to choose $\gamma = 1$ and

$$G_k = \left(\frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\phi(i_t)'\right)^{-1},$$

[cf. Eq. (6.73)]. Diagonal approximations to this matrix may also be used to avoid the computational overhead of matrix inversion. Another possibility is

$$G_k = \Big(C_k^{(\lambda)'} \Sigma_k^{-1} C_k^{(\lambda)} + \beta I\Big)^{-1} C_k^{(\lambda)'} \Sigma_k^{-1}, \tag{6.90}$$

where Σ_k is some positive definite symmetric matrix, and β is a positive scalar [cf. Eq. (6.75)]. For $\gamma = 1$, we obtain the iteration

$$r_{k+1} = \left(C_k^{(\lambda)'}\Sigma_k^{-1}C_k^{(\lambda)} + \beta I\right)^{-1}\left(C_k^{(\lambda)'}\Sigma_k^{-1}d_k^{(\lambda)} + \beta r_k\right). \tag{6.91}$$

This as an iterative version of the regression-based LSTD method [cf. Eq. (6.76)], for which convergence is assured provided $C_k^{(\lambda)} \to C^{(\lambda)}$, $d_k^{(\lambda)} \to d^{(\lambda)}$, and $\{\Sigma_k^{-1}\}$ is bounded.

Regarding the calculation of appropriate simulation-based approximations $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$, one possibility is the following extension of Eqs. (6.48)-(6.49):

$$C_k^{(\lambda)} = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\sum_{m=t}^{k}\alpha^{m-t}\lambda^{m-t}\big(\phi(i_m) - \alpha\phi(i_{m+1})\big)', \tag{6.92}$$

$$d_k^{(\lambda)} = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\sum_{m=t}^{k}\alpha^{m-t}\lambda^{m-t}g_{i_m}. \tag{6.93}$$

It can be shown that indeed these are correct simulation-based approximations to $C^{(\lambda)}$ and $d^{(\lambda)}$ of Eq. (6.86). The verification is similar to the case $\lambda = 0$, by considering the approximation of the steady-state probabilities ξ_i and transition probabilities p_{ij} with the empirical frequencies $\hat{\xi}_{i,k}$ and $\hat{p}_{ij,k}$ defined by Eq. (6.68).

For a sketch of the argument, we first verify that the rightmost expression in the definition (6.92) of $C_k^{(\lambda)}$ can be written as

$$\sum_{m=t}^{k}\alpha^{m-t}\lambda^{m-t}\big(\phi(i_m) - \alpha\phi(i_{m+1})\big)'$$
$$= \phi(i_t) - \alpha(1-\lambda)\sum_{m=t}^{k-1}\alpha^{m-t}\lambda^{m-t}\phi(i_{m+1}) - \alpha^{k-t+1}\lambda^{k-t}\phi(i_{k+1}),$$

which by discarding the last term (it is negligible for $k >> t$), yields

$$\sum_{m=t}^{k}\alpha^{m-t}\lambda^{m-t}\big(\phi(i_m) - \alpha\phi(i_{m+1})\big)'$$
$$= \phi(i_t) - \alpha(1-\lambda)\sum_{m=t}^{k-1}\alpha^{m-t}\lambda^{m-t}\phi(i_{m+1}).$$

Using this relation in the expression (6.92) for $C_k^{(\lambda)}$, we obtain

$$C_k^{(\lambda)} = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\left(\phi(i_t) - \alpha(1-\lambda)\sum_{m=t}^{k-1}\alpha^{m-t}\lambda^{m-t}\phi(i_{m+1})\right)'.$$

We now compare this expression with $C^{(\lambda)}$, which similar to Eq. (6.50), can be written as

$$C^{(\lambda)} = \Phi' \Xi (I - \alpha P^{(\lambda)}) \Phi = \sum_{i=1}^{n} \xi_i \phi(i) \left(\phi(i) - \alpha \sum_{j=1}^{n} p_{ij}^{(\lambda)} \phi(j) \right)',$$

where $p_{ij}^{(\lambda)}$ are the components of the matrix $P^{(\lambda)}$. It can be seen (cf. the derivations of Section 6.3.3) that

$$\frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \phi(i_t)' \to \sum_{i=1}^{n} \xi_i \phi(i) \phi(i)',$$

while by using the formula

$$p_{ij}^{(\lambda)} = (1-\lambda) \sum_{\ell=0}^{\infty} \alpha^{\ell} \lambda^{\ell} p_{ij}^{(\ell+1)}$$

with $p_{ij}^{(\ell+1)}$ being the (i,j)th component of $P^{(\ell+1)}$ [cf. Eq. (6.85)], it can be verified that

$$\frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \left((1-\lambda) \sum_{m=t}^{k-1} \alpha^{m-t} \lambda^{m-t} \phi(i_{m+1})' \right) \to \sum_{i=1}^{n} \xi_i \phi(i) \sum_{j=1}^{n} p_{ij}^{(\lambda)} \phi(j)'.$$

Thus, by comparing the preceding expressions, we see that $C_k^{(\lambda)} \to C^{(\lambda)}$ with probability 1. A full convergence analysis can be found in [NeB03] and also in [BeY09], [Yu10a,b], in a more general exploration-related context, to be discussed in Section 6.3.7 and also in Section 6.8.

We may also streamline the calculation of $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ by introducing the vector

$$z_t = \sum_{m=0}^{t} (\alpha\lambda)^{t-m} \phi(i_m), \tag{6.94}$$

which is often called the *eligibility vector*; it is a weighted sum of the present and past feature vectors $\phi(i_m)$ obtained from the simulations [discounted by $(\alpha\lambda)^{t-m}$]. Then, by straightforward calculation, we may verify that

$$C_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^{k} z_t \big(\phi(i_t) - \alpha \phi(i_{t+1}) \big)', \tag{6.95}$$

$$d_k^{(\lambda)} = \frac{1}{k+1} \sum_{t=0}^{k} z_t \, g(i_t, i_{t+1}). \tag{6.96}$$

Note that z_k, $C_k^{(\lambda)}$, $d_k^{(\lambda)}$, can be conveniently updated by means of recursive formulas, as in the case $\lambda = 0$. In particular, we have

$$z_k = \alpha\lambda\, z_{k-1} + \phi(i_k),$$

$$C_k^{(\lambda)} = (1-\delta_k)C_{k-1}^{(\lambda)} + \delta_k z_k\big(\phi(i_k) - \alpha\phi(i_{k+1})\big)',$$

$$d_k^{(\lambda)} = (1-\delta_k)d_{k-1}^{(\lambda)} + \delta_k z_k g(i_k, i_{k+1}),$$

with the initial conditions $z_{-1} = 0$, $C_{-1} = 0$, $d_{-1} = 0$, and

$$\delta_k = \frac{1}{k+1}, \quad k = 0, 1, \ldots.$$

Let us also note that by using the above formulas for $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$, the scaled LSPE(λ) iteration (6.89) can also be written as

$$r_{k+1} = r_k - \frac{\gamma}{k+1} G_k \sum_{t=0}^{k} z_t q_{k,t}, \tag{6.97}$$

where $q_{k,t}$ is the temporal difference

$$q_{k,t} = \phi(i_t)'r_k - \alpha\phi(i_{t+1})'r_k - g(i_t, i_{t+1}) \tag{6.98}$$

[cf. Eqs. (6.54) and (6.71)].

The TD(λ) algorithm is essentially TD(0) applied to the multistep projected equation $C^{(\lambda)}r = d^{(\lambda)}$. It takes the form

$$r_{k+1} = r_k - \gamma_k z_k q_{k,k}, \tag{6.99}$$

where γ_k is a stepsize parameter. When compared to the scaled LSPE(λ) method (6.97), we see that TD(λ) uses $G_k = I$ and only the latest temporal difference $q_{k,k}$. This amounts to approximating $C^{(\lambda)}$ and $d^{(\lambda)}$ by a single sample, instead of $k+1$ samples. Note that as $\lambda \to 1$, z_k approaches $\sum_{t=0}^{k} \alpha^{k-t}\phi(i_t)$ [cf. Eq. (6.94)], and TD(λ) approaches the TD(1) method given earlier in Section 6.2 [cf. Eq. (6.32)].

Least Squares Implementation of LSPE(λ)

Let us now discuss an alternative development of the (unscaled) LSPE(λ) method, which is based on the PVI(λ) method and parallels the implementation (6.64) for LSPE(0). We first obtain an alternative formula for $T^{(\lambda)}$, and to this end we view $T^{t+1}J$ as the vector of costs over a horizon of $(t+1)$ stages with the terminal cost function being J, and write

$$T^{t+1}J = \alpha^{t+1}P^{t+1}J + \sum_{k=0}^{t} \alpha^k P^k g. \tag{6.100}$$

As a result the mapping $T^{(\lambda)} = (1-\lambda)\sum_{t=0}^{\infty}\lambda^t T^{t+1}$ can be expressed as

$$(T^{(\lambda)}J)(i) = \sum_{t=0}^{\infty}(1-\lambda)\lambda^t E\left\{\alpha^{t+1}J(i_{t+1}) + \sum_{k=0}^{t}\alpha^k g(i_k, i_{k+1}) \;\Big|\; i_0 = i\right\}, \tag{6.101}$$

which can be written as

$$\begin{aligned}(T^{(\lambda)}J)(i) &= J(i) + (1-\lambda)\\ &\qquad\cdot\sum_{t=0}^{\infty}\sum_{k=0}^{t}\lambda^t\alpha^k E\big\{g(i_k, i_{k+1}) + \alpha J_t(i_{k+1}) - J_t(i_k) \mid i_0 = i\big\}\\ &= J(i) + (1-\lambda)\\ &\qquad\cdot\sum_{k=0}^{\infty}\left(\sum_{t=k}^{\infty}\lambda^t\right)\alpha^k E\big\{g(i_k, i_{k+1}) + \alpha J(i_{k+1}) - J(i_k) \mid i_0 = i\big\}\end{aligned}$$

and finally,

$$(T^{(\lambda)}J)(i) = J(i) + \sum_{t=0}^{\infty}(\alpha\lambda)^k E\big\{g(i_t, i_{t+1}) + \alpha J(i_{t+1}) - J(i_t) \mid i_0 = i\big\}.$$

Using this equation, we can write the PVI(λ) iteration

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

as

$$\begin{aligned}r_{k+1} = \arg\min_{r\in\Re^s}\sum_{i=1}^{n}\xi_i\Bigg(&\phi(i)'r - \phi(i)'r_k\\ &- \sum_{t=0}^{\infty}(\alpha\lambda)^t E\big\{g(i_t, i_{t+1}) + \alpha\phi(i_{t+1})'r_k - \phi(i_t)'r_k \mid i_0 = i\big\}\Bigg)^2\end{aligned}$$

and by introducing the temporal differences

$$d_k(i_t, i_{t+1}) = g(i_t, i_{t+1}) + \alpha\phi(i_{t+1})'r_k - \phi(i_t)'r_k,$$

we finally obtain PVI(λ) in the form

$$\begin{aligned}r_{k+1} = \arg\min_{r\in\Re^s}\sum_{i=1}^{n}\xi_i\Bigg(&\phi(i)'r - \phi(i)'r_k\\ &- \sum_{t=0}^{\infty}(\alpha\lambda)^t E\big\{d_k(i_t, i_{t+1}) \mid i_0 = i\big\}\Bigg)^2.\end{aligned} \tag{6.102}$$

The LSPE(λ) method is a simulation-based approximation to the above PVI(λ) iteration. It has the form

$$r_{k+1} = \arg\min_{r\in\Re^s} \sum_{t=0}^{k} \left(\phi(i_t)'r - \phi(i_t)'r_k - \sum_{m=t}^{k} (\alpha\lambda)^{m-t} d_k(i_m, i_{m+1}) \right)^2, \tag{6.103}$$

where $(i_0, i_1, \ldots)$ is an infinitely long trajectory generated by simulation. The justification is that the solution of the least squares problem in the PVI(λ) iteration (6.102) is approximately equal to the solution of the least squares problem in the LSPE(λ) iteration (6.103). Similar to the case $\lambda = 0$ [cf. Eqs. (6.63) and (6.64)], the approximation is due to:

(a) The substitution of the steady-state probabilities ξ_i and transition probabilities p_{ij} with the empirical frequencies $\hat{\xi}_{i,k}$ and $\hat{p}_{ij,k}$ defined by Eq. (6.68).

(b) The approximation of the infinite discounted sum of temporal differences in Eq. (6.102) with the finite discounted sum in Eq. (6.103), which also uses an approximation of the conditional probabilities of the transitions (i_t, i_{t+1}) with corresponding empirical frequencies.

Since as $k \to \infty$, the empirical frequencies converge to the true probabilities and the finite discounted sums converge to the infinite discounted sums, it follows that PVI(λ) and LSPE(λ) asymptotically coincide.

Exploration-Enhanced LSPE(λ), LSTD(λ), and TD(λ)

We next develop an alternative least squares implementation of LSPE(λ). It uses multiple simulation trajectories and the initial state of each trajectory may be chosen essentially as desired, thereby allowing flexibility to generate a richer mixture of state visits. In particular, we generate t simulated trajectories. The states of a trajectory are generated according to the transition probabilities p_{ij} of the policy under evaluation, the transition cost is discounted by an additional factor α with each transition, and following each transition to a state j, the trajectory is terminated with probability $1 - \lambda$ and with an extra cost $\alpha\phi(i)'r_k$, where Φr_k is the current estimate of the cost vector of the policy under evaluation. Once a trajectory is terminated, an initial state for the next trajectory is chosen according to a fixed probability distribution $\zeta_0 = \big(\zeta_0(1), \ldots, \zeta_0(n)\big)$, where

$$\zeta_0(i) = P(i_0 = i) > 0, \qquad i = 1, \ldots, n,$$

and the process is repeated. The details are as follows.

Let the mth trajectory have the form $(i_{0,m}, i_{1,m}, \ldots, i_{N_m,m})$, where $i_{0,m}$ is the initial state, and $i_{N_m,m}$ is the state at which the trajectory

is completed (the last state prior to termination). For each state $i_{\ell,m}$, $\ell = 0, \ldots, N_m - 1$, of the mth trajectory, the simulated cost is

$$c_{\ell,m}(r_k) = \alpha^{N_m-\ell}\phi(i_{N_m,m})'r_k + \sum_{q=\ell}^{N_m-1} \alpha^{q-\ell} g(i_{q,m}, i_{q+1,m}). \tag{6.104}$$

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

$$r_{k+1} = \arg\min_{r\in\Re^s} \sum_{m=1}^{t} \sum_{\ell=0}^{N_m-1} \big(\phi(i_{\ell,m})'r - c_{\ell,m}(r_k)\big)^2, \tag{6.105}$$

similar to Eq. (6.103).

We will now show that in the limit, as $t \to \infty$, the vector r_{k+1} of Eq. (6.105) satisfies

$$\Phi r_{k+1} = \hat{\Pi} T^{(\lambda)}(\Phi r_k), \tag{6.106}$$

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

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

and $\hat{\zeta}(i) = \sum_{\ell=0}^{\infty} \zeta_\ell(i)$, with $\zeta_\ell(i)$ being the probability of the state being i after ℓ steps of a randomly chosen simulation trajectory. Note that $\zeta(i)$ is the long-term occupancy probability of state i during the simulation process.

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

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

where P and g are the transition probability matrix and cost vector, respectively, under the current policy. As a result the vector $T^{(\lambda)}J = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}J$ can be expressed as

$$\big(T^{(\lambda)}J\big)(i) = \sum_{\ell=0}^{\infty}(1-\lambda)\lambda^\ell E\left\{\alpha^{\ell+1}J(i_{\ell+1}) + \sum_{q=0}^{\ell}\alpha^q g(i_q, i_{q+1}) \;\Big|\; i_0 = i\right\}. \tag{6.107}$$

Thus $\big(T^{(\lambda)}J\big)(i)$ may be viewed as the expected value of the $(\ell+1)$-stages cost of the policy under evaluation starting at state i, with the number

of stages being random and geometrically distributed with parameter λ [probability of $\kappa+1$ transitions is $(1-\lambda)\lambda^\kappa$, $\kappa=0,1,\ldots$]. It follows that the cost samples $c_{\ell,m}(r_k)$ of Eq. (6.104), produced by the simulation process described earlier, can be used to estimate $\big(T^{(\lambda)}(\Phi r_k)\big)(i)$ for all i by Monte Carlo averaging. The estimation formula is

$$C_t(i)=\frac{1}{\sum_{m=1}^{t}\sum_{\ell=0}^{N_m-1}\delta(i_{\ell,m}=i)}\cdot\sum_{m=1}^{t}\sum_{\ell=0}^{N_m-1}\delta(i_{\ell,m}=i)c_{\ell,m}(r_k), \quad (6.108)$$

where for any event E, we denote by $\delta(E)$ the indicator function of E, and we have

$$\big(T^{(\lambda)}(\Phi r_k)\big)(i)=\lim_{t\to\infty}C_t(i), \qquad i=1,\ldots,n,$$

(see also the discussion on the consistency of Monte Carlo simulation for policy evaluation in Section 6.2, Exercise 6.2, and [BeT96], Section 5.2).

Let us now compare iteration (6.106) with the simulation-based implementation (6.105). Using the definition of projection, Eq. (6.106) can be written as

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

or equivalently

$$r_{k+1}=\left(\sum_{i=1}^{n}\zeta(i)\phi(i)\phi(i)'\right)^{-1}\sum_{i=1}^{n}\zeta(i)\phi(i)\big(T^{(\lambda)}(\Phi r_k)\big)(i). \quad (6.109)$$

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

$$\tilde{\zeta}(i)=\frac{1}{N_1+\cdots+N_t}\sum_{m=1}^{t}\sum_{\ell=0}^{N_m-1}\delta(i_{\ell,m}=i). \quad (6.110)$$

Then the simulation-based estimate (6.105) can be written as

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

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

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

and finally, using Eqs. (6.108) and (6.110),

$$r_{k+1} = \left(\sum_{i=1}^{n} \tilde{\zeta}(i)\phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^{n} \tilde{\zeta}(i)\phi(i)C_t(i). \tag{6.111}$$

Since $\big(T^{(\lambda)}(\Phi r_k)\big)(i) = \lim_{t\to\infty} C_t(i)$ and $\zeta(i) = \lim_{t\to\infty} \tilde{\zeta}(i)$, we see that the iteration (6.109) and the simulation-based implementation (6.111) asymptotically coincide.

An important fact is that the implementation just described deals effectively with the issue of exploration. Since each simulation trajectory is completed at each transition with the potentially large probability $1-\lambda$, a restart with a new initial state i_0 is frequent and the length of each of the simulated trajectories is relatively small. Thus the restart mechanism can be used as a "natural" form of exploration, by choosing appropriately the restart distribution ζ_0, so that $\zeta_0(i)$ reflects a "substantial" weight for all states i.

An interesting special case is when $\lambda = 0$, in which case the simulated trajectories consist of a single transition. Thus there is a restart at every transition, which means that the simulation samples are from states that are generated independently according to the restart distribution ζ_0.

We can also develop similarly, a least squares exploration-enhanced implementation of LSTD(λ). We use the same simulation procedure, and in analogy to Eq. (6.104) we define

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

The LSTD(λ) approximation $\Phi\hat{r}$ to the projected equation

$$\Phi r = \hat{\Pi} T^{(\lambda)}(\Phi r),$$

[cf. Eq. (6.106)] is determined from the fixed point equation

$$\hat{r} = \arg\min_{r\in\Re^s} \sum_{m=1}^{t} \sum_{\ell=0}^{N_m-1} \big(\phi(i_{\ell,m})'r - c_{\ell,m}(\hat{r})\big)^2. \tag{6.112}$$

By writing the optimality condition

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

for the above least squares minimization and solving for $\hat{r}$, we obtain

$$\hat{r} = \hat{C}^{-1}\hat{d}, \tag{6.113}$$

where

$$\hat{C} = \sum_{m=1}^{t} \sum_{\ell=0}^{N_m-1} \phi(i_{\ell,m})\big(\phi(i_{\ell,m}) - \alpha^{N_m-\ell}\phi(i_{N_m,m})\big)', \tag{6.114}$$

and

$$\hat{d} = \sum_{m=1}^{t} \sum_{\ell=0}^{N_m-1} \sum_{q=\ell}^{N_m-1} \alpha^{q-\ell} g(i_{q,m}, i_{q+1,m}). \tag{6.115}$$

For a large number of trajectories t, the methods (6.105) and (6.112) [or equivalently (6.113)-(6.115)] yield similar results, particularly when $\lambda \approx 1$. However, the method (6.105) has an iterative character (r_{k+1} depends on r_k), so it is reasonable to expect that it is less susceptible to simulation noise in an optimistic PI setting where the number of samples per policy is low.

Similarly, to obtain an exploration-enhanced TD(λ), we simply solve approximately the least squares problem in Eq. (6.105) by iterating, perhaps multiple times, with an incremental gradient method. The details of this type of algorithm are straightforward (see Section 6.2). The method does not involve matrix inversion like the exploration-enhanced implementations (6.105) and (6.113)-(6.115) of LSPE(λ) and LSTD(λ), respectively, but is much slower and less reliable.

Feature Scaling and its Effect on LSTD(λ), LSPE(λ), and TD(λ)

Let us now discuss how the representation of the approximation subspace S affects the results produced by LSTD(λ), LSPE(λ), and TD(λ). In particular, suppose that instead of S being represented as

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

it is equivalently represented as

$$S = \{\Psi v \mid v \in \Re^r\},$$

where

$$\Phi = \Psi B,$$

with B being an invertible $r \times r$ matrix. Thus S is represented as the span of a different set of basis functions, and any vector $\Phi r \in S$ can be written as Ψv, where the weight vector v is equal to Br. Moreover, each row $\phi(i)'$, the feature vector of state i in the representation based on Φ, is equal to $\psi(i)'B$, the linearly transformed feature vector of i in the representation based on Ψ.

Suppose that we generate a trajectory $(i_0, i_1, \ldots)$ according to the simulation process of Section 6.3.3, and we calculate the iterates of LSTD(λ), LSPE(λ), and TD(λ) using the two different representations of S, based on Φ and Ψ. Let $C_{k,\Phi}^{(\lambda)}$ and $C_{k,\Psi}^{(\lambda)}$ be the corresponding matrices generated by Eq. (6.95), and let $d_{k,\Phi}^{(\lambda)}$ and $d_{k,\Psi}^{(\lambda)}$ be the corresponding vectors generated by Eq. (6.96). Let also $z_{t,\Phi}$ and $z_{t,\Psi}$ be the corresponding eligibility vectors generated by Eq. (6.94). Then, since $\phi(i_m) = B'\psi(i_m)$, we have

$$z_{t,\Phi} = B' z_{t,\Psi},$$

and from Eqs. (6.95) and (6.96),

$$C_{k,\Phi}^{(\lambda)} = B' C_{k,\Psi}^{(\lambda)} B, \qquad d_{k,\Phi}^{(\lambda)} = B' d_{k,\Psi}^{(\lambda)}.$$

We now wish to compare the high dimensional iterates Φr_k and Ψv_k produced by different methods. Based on the preceding equation, we claim that *LSTD(λ) is scale-free* in the sense that $\Phi r_k = \Psi v_k$ for all k. Indeed, in the case of LSTD(λ) we have [cf. Eq. (6.88)]

$$\Phi r_k = \Phi \big(C_{k,\Phi}^{(\lambda)}\big)^{-1} d_{k,\Phi}^{(\lambda)} = \Psi B \big(B' C_{k,\Psi}^{(\lambda)} B\big)^{-1} B' d_{k,\Psi}^{(\lambda)} = \Psi \big(C_{k,\Psi}^{(\lambda)}\big)^{-1} d_{k,\Psi}^{(\lambda)} = \Psi v_k.$$

We also claim that *LSPE(λ) with*

$$G_k = \left(\frac{1}{k+1}\sum_{t=0}^{k} \phi(i_t)\phi(i_t)'\right)^{-1},$$

[cf. Eq. (6.73)], is scale-free in the sense that $\Phi r_k = \Psi v_k$ for all k. This follows from Eq. (6.97) using a calculation similar to the one for LSTD(λ), but it also follows intuitively from the fact that LSPE(λ), with G_k as given above, is a simulation-based implementation of the PVI(λ) iteration $J_{k+1} = \Pi T J_k$, which involves the projection operator Π that is scale-free (does not depend on the representation of S).

We finally note that *the TD(λ) iteration (6.99) is not scale-free* unless B is an orthogonal matrix ($BB' = I$). This can be verified with a direct calculation using the iteration (6.99) for the case of the two representations of S based on Φ and Ψ. In particular, let $\{r_k\}$ be generated by TD(λ) based on Φ,

$$r_{k+1} = r_k - \gamma_k z_{k,\Phi}\big(\phi(i_k)'r_k - \alpha\phi(i_{k+1})'r_k - g(i_k, i_{k+1})\big),$$

and let $\{v_k\}$ be generated by TD(λ) based on Ψ,

$$v_{k+1} = v_k - \gamma_k z_{k,\Psi}\big(\psi(i_k)'v_k - \alpha\psi(i_{k+1})'v_k - g(i_k, i_{k+1})\big),$$

[cf. Eqs. (6.98), (6.99)]. Then, we generally have $\Phi r_k \neq \Psi v_k$, since $\Phi r_k = \Psi B r_k$ and $B r_k \neq v_k$. In particular, the vector $\overline{v}_k = B r_k$ is generated by the iteration

$$\overline{v}_{k+1} = \overline{v}_k - \gamma_k z_{k,\Psi}(BB')\big(\psi(i_k)'\overline{v}_k - \alpha\psi(i_{k+1})'\overline{v}_k - g(i_k, i_{k+1})\big),$$

which is different from the iteration that generates v_k, unless $BB' = I$. This analysis also indicates that the appropriate value of the stepsize γ_k in TD(λ) strongly depends on the choice of basis functions to represent S, and points to a generic weakness of the method.

6.3.7 Policy Iteration Issues – Exploration

We have discussed so far policy evaluation methods based on the projected equation. We will now address, in this and the next subsection, some of the difficulties associated with these methods, when embedded within policy iteration. One difficulty has to do with the issue of exploration: for a variety of reasons it is important to generate trajectories according to the steady-state distribution ξ associated with the given policy μ (one reason is the need to concentrate on "important" states that are likely to occur under a near-optimal policy, and another is the desirabiity to maintain the contraction property of ΠT). On the other hand, this biases the simulation by underrepresenting states that are unlikely to occur under μ, causing potentially serious errors in the calculation of a new policy via policy improvement.

Another difficulty is that Assumption 6.3.1 (the irreducibility of the transition matrix P of the policy being evaluated) may be hard or impossible to guarantee, in which case the methods break down, either because of the presence of transient states (in which case the components of ξ corresponding to transient states are 0, and these states are not represented in the constructed approximation), or because of multiple recurrent classes (in which case some states will never be generated during the simulation, and again will not be represented in the constructed approximation).

We noted earlier one possibility to introduce a natural form of exploration in a least squares implementation of LSPE(λ) and LSTD(λ). We will now discuss another popular approach to address the exploration difficulty, which is often used in conjunction with LSTD. This is to modify the transition matrix P of the given policy μ by occasionally generating transitions other than the ones dictated by μ. If the modified transition probability matrix is irreducible, we simultaneously address the difficulty with multiple recurrent classes and transient states as well. Mathematically, in such a scheme we generate an infinitely long trajectory $(i_0, i_1, \ldots)$ according to an irreducible transition probability matrix

$$\overline{P} = (I - B)P + BQ, \tag{6.116}$$

where B is a diagonal matrix with diagonal components $\beta_i \in [0, 1]$ and Q is another transition probability matrix. Thus, at state i, the next state is generated with probability $1 - \beta_i$ according to transition probabilities p_{ij}, and with probability β_i according to transition probabilities q_{ij} [here pairs (i, j) with $q_{ij} > 0$ need not correspond to physically plausible transitions].† We refer to β_i as the *exploration probability* at state i.

Unfortunately, using $\overline{P}$ in place of P for simulation, with no other modification in the TD algorithms, creates a bias that tends to degrade the quality of policy evaluation, because it directs the algorithms towards approximating the fixed point of the mapping $\overline{T}^{(\lambda)}$, given by

$$\overline{T}^{(\lambda)}(J) = \overline{g}^{(\lambda)} + \alpha \overline{P}^{(\lambda)} J,$$

where

$$\overline{T}^{(\lambda)}(J) = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t \overline{T}^{t+1}(J),$$

with

$$\overline{T}(J) = \overline{g} + \alpha \overline{P} J$$

[cf. Eq. (6.85)]. This is the cost of a *different* policy, a fictitious exploration-enhanced policy that has a cost vector $\overline{g}$ with components

$$\overline{g}_i = \sum_{j=1}^{n} \overline{p}_{ij} g(i, j), \qquad i = 1, \ldots, n,$$

† In the literature, e.g., [SuB98], the policy being evaluated is sometimes called the *target policy* to distinguish from a policy modified for exploration like $\overline{P}$, which is called *behavior policy*. Also, methods that use a behavior policy are called *off-policy* methods, while methods that do not are called *on-policy* methods. Note, however, that $\overline{P}$ need not correspond to an admissible policy, and indeed there may not exist a suitable admissible policy that can induce sufficient exploration.

and a transition probability matrix $\overline{P}$ in place of P. In particular, when the simulated trajectory is generated according to $\overline{P}$, the LSTD(λ), LSPE(λ), and TD(λ) algorithms yield the unique solution $\overline{r}_\lambda$ of the equation

$$\Phi r = \overline{\Pi}\,\overline{T}^{(\lambda)}(\Phi r), \tag{6.117}$$

where $\overline{\Pi}$ denotes projection on the approximation subspace with respect to $\|\cdot\|_{\overline{\xi}}$, where $\overline{\xi}$ is the invariant distribution corresponding to $\overline{P}$.

We will discuss in this section some schemes that allow the approximation of the solution of the projected equation

$$\Phi r = \overline{\Pi} T^{(\lambda)}(\Phi r), \tag{6.118}$$

where $\overline{\Pi}$ is projection with respect to the norm $\|\cdot\|_{\overline{\xi}}$, corresponding to the steady-state distribution $\overline{\xi}$ of $\overline{P}$. Note the difference between equations (6.117) and (6.118): *the first involves $\overline{T}$ but the second involves T, so it aims to approximate the desired fixed point of T, rather than a fixed point of $\overline{T}$.* Thus, the following schemes allow exploration, but without the degradation of approximation quality resulting from the use of $\overline{T}$ in place of T.

Exploration Using Extra Transitions

The first scheme applies only to the case where $\lambda = 0$. Then a vector r^* solves the exploration-enhanced projected equation $\Phi r = \overline{\Pi} T(\Phi r)$ if and only if it satisfies the orthogonality condition

$$\Phi' \overline{\Xi}(\Phi r^* - \alpha P \Phi r^* - g) = 0, \tag{6.119}$$

where $\overline{\Xi}$ is the steady-state distribution of $\overline{P}$ [cf. Eq. (6.39)]. This condition can be written in matrix form as

$$Cr^* = d,$$

where

$$C = \Phi' \overline{\Xi}(I - \alpha P)\Phi, \qquad d = \Phi' \overline{\Xi} g. \tag{6.120}$$

These equations should be compared with the equations for the case where $\overline{P} = P$ [cf. Eqs. (6.40)-(6.41)]: the only difference is that the distribution matrix Ξ is replaced by the exploration-enhanced distribution matrix $\overline{\Xi}$.

We generate a state sequence $\{i_0, i_1, \ldots\}$ according to the exploration-enhanced transition matrix $\overline{P}$ (or in fact any steady state distribution $\overline{\xi}$, such as the uniform distribution). We also generate an *additional* sequence of independent transitions $\{(i_0, j_0), (i_1, j_1), \ldots\}$ according to the original transition matrix P.

We approximate the matrix C and vector d of Eq. (6.120) using the formulas

$$C_k = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\big(\phi(i_t)-\alpha\phi(j_t)\big)',$$

and

$$d_k = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)g(i_t,j_t),$$

in place of Eqs. (6.48) and (6.49). Similar to the earlier case in Section 6.3.3, where $\overline{P}=P$, it can be shown using law of large numbers arguments that $C_k \to C$ and $d_k \to d$ with probability 1.

The corresponding approximation $C_k r = d_k$ to the projected equation $\Phi r = \overline{\Pi}T(\Phi r)$ can be written as

$$\sum_{t=0}^{k}\phi(i_t)\tilde{q}_{k,t}=0, \tag{6.121}$$

where

$$\tilde{q}_{k,t}=\phi(i_t)'r_k-\alpha\phi(j_t)'r_k-g(i_t,j_t)$$

is a temporal difference associated with the transition (i_t,j_t) [cf. Eq. (6.54)]. The three terms in the definition of $\tilde{q}_{k,t}$ can be viewed as samples [associated with the transition (i_t,j_t)] of the corresponding three terms of the expression $\overline{\Xi}(\Phi r_k-\alpha P\Phi r_k-g)$ in Eq. (6.119).

In a modified form of LSTD(0), we approximate the solution $C^{-1}d$ of the projected equation with $C_k^{-1}d_k$. In a modified form of (scaled) LSPE(0), we approximate the term (Cr_k-d) in PVI by $(C_kr_k-d_k)$, leading to the iteration

$$r_{k+1}=r_k-\frac{\gamma}{k+1}G_k\sum_{t=0}^{k}\phi(i_t)\tilde{q}_{k,t}, \tag{6.122}$$

where γ is small enough to guarantee convergence [cf. Eq. (6.72)]. Finally, the modified form of TD(0) is

$$r_{k+1}=r_k-\gamma_k\phi(i_k)\tilde{q}_{k,k}, \tag{6.123}$$

where γ_k is a positive diminishing stepsize [cf. Eq. (6.78)]. Unfortunately, versions of these schemes for $\lambda>0$ are complicated because of the difficulty of generating extra transitions in a multistep context.

Exploration Using Modified Temporal Differences

We will now present an alternative exploration approach that works for all $\lambda \geq 0$. Like the preceding approach, it aims to solve the projected equation $\Phi r = \overline{\Pi} T^{(\lambda)}(\Phi r)$ [cf. Eq. (6.118)], but it does not require extra transitions. It does require, however, the explicit knowledge of the transition probabilities p_{ij} and $\overline{p}_{ij}$, so it does not apply to the model-free context. (Later, in Section 6.5, we will discuss appropriate model-free modifications in the context of Q-learning.)

Here we generate a single state sequence $\{i_0, i_1, \ldots\}$ according to the exploration-enhanced transition matrix $\overline{P}$. The formulas of the various TD algorithms are similar to the ones given earlier, but we use modified versions of temporal differences, defined by

$$\tilde{q}_{k,t} = \phi(i_t)' r_k - \frac{p_{i_t i_{t+1}}}{\overline{p}_{i_t i_{t+1}}} \big(\alpha \phi(i_{t+1})' r_k + g(i_t, i_{t+1})\big), \tag{6.124}$$

where p_{ij} and $\overline{p}_{ij}$ denote the ijth components of P and $\overline{P}$, respectively.†

Consider now the case where $\lambda = 0$ and the approximation of the matrix C and vector d of Eq. (6.120) by simulation: we generate a state sequence $\{i_0, i_1, \ldots\}$ using the exploration-enhanced transition matrix $\overline{P}$. After collecting $k+1$ samples ($k = 0, 1, \ldots$), we form

$$C_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \left(\phi(i_t) - \alpha \frac{p_{i_t i_{t+1}}}{\overline{p}_{i_t i_{t+1}}} \phi(i_{t+1}) \right)',$$

and

$$d_k = \frac{1}{k+1} \sum_{t=0}^{k} \frac{p_{i_t i_{t+1}}}{\overline{p}_{i_t i_{t+1}}} \phi(i_t) g(i_t, i_{t+1}).$$

Similar to the earlier case in Section 6.3.3, where $\overline{P} = P$, it can be shown using simple law of large numbers arguments that $C_k \to C$ and $d_k \to d$ with probability 1 (see also Section 6.8.1, where this approximation approach is discussed within a more general context). Note that the approximation $C_k r = d_k$ to the projected equation can also be written as

$$\sum_{t=0}^{k} \phi(i_t) \tilde{q}_{k,t} = 0,$$

† Note the difference in the sampling of transitions. Whereas in the preceding scheme with extra transitions, (i_t, j_t) was generated according to the original transition matrix P, here (i_t, i_{t+1}) is generated according to the exploration-enhanced transition matrix $\overline{P}$. The approximation of an expected value with respect to a given distribution (induced by the transition matrix P) by sampling with respect to a different distribution (induced by the exploration-enhanced transition matrix $\overline{P}$) is reminiscent of importance sampling (cf. Section 6.1.5). The probability ratio $\frac{p_{i_t i_{t+1}}}{\overline{p}_{i_t i_{t+1}}}$ in Eq. (6.124) provides the necessary correction.

where $\tilde{q}_{k,t}$ is the modified temporal difference given by Eq. (6.124) [cf. Eq. (6.121)].

The exploration-enhanced LSTD(0) method is simply $\hat{r}_k = C_k^{-1} d_k$, and converges with probability 1 to the solution of the projected equation $\Phi r = \overline{\Pi} T(\Phi r)$. Exploration-enhanced versions of LSPE(0) and TD(0) can be similarly derived [cf. Eqs. (6.122) and (6.123)], but for convergence of these methods, the mapping $\overline{\Pi} T$ should be a contraction, which is guaranteed only if $\overline{P}$ differs from P by a small amount (see the subsequent Prop. 6.3.6).

Let us now consider the case where $\lambda > 0$. We first note that increasing values of λ tend to preserve the contraction of $\overline{\Pi} T^{(\lambda)}$. In fact, given any norm $\|\cdot\|_{\overline{\xi}}$, $T^{(\lambda)}$ is a contraction with respect to that norm, provided λ is sufficiently close to 1 (see Prop. 6.3.5, which shows that the contraction modulus of $T^{(\lambda)}$ tends to 0 as $\lambda \to 1$). This implies that given any exploration probabilities from the range $[0,1]$ such that $\overline{P}$ is irreducible, there exists $\overline{\lambda} \in [0,1)$ such that $T^{(\lambda)}$ and $\overline{\Pi} T^{(\lambda)}$ are contractions with respect to $\|\cdot\|_{\overline{\xi}}$ for all $\lambda \in [\overline{\lambda}, 1)$.

Exploration-enhanced versions of LSTD(λ) and LSPE(λ) have been obtained by Bertsekas and Yu [BeY09], to which we refer for their detailed development. In particular, the exploration-enhanced LSTD(λ) method computes $\hat{r}_k$ as the solution of the equation $C_k^{(\lambda)} r = d_k^{(\lambda)}$, with $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ generated with recursions similar to the ones with unmodified TD [cf. Eqs. (6.94)-(6.96)]:

$$C_k^{(\lambda)} = (1-\delta_k) C_{k-1}^{(\lambda)} + \delta_k z_k \left(\phi(i_k) - \alpha \frac{p_{i_k i_{k+1}}}{\overline{p}_{i_k i_{k+1}}} \phi(i_{k+1}) \right)', \tag{6.125}$$

$$d_k^{(\lambda)} = (1-\delta_k) d_{k-1}^{(\lambda)} + \delta_k z_k \frac{p_{i_k i_{k+1}}}{\overline{p}_{i_k i_{k+1}}} g(i_k, i_{k+1}), \tag{6.126}$$

where z_k are modified eligibility vectors given by

$$z_k = \alpha\lambda \frac{p_{i_{k-1} i_k}}{\overline{p}_{i_{k-1} i_k}} z_{k-1} + \phi(i_k), \tag{6.127}$$

the initial conditions are $z_{-1} = 0$, $C_{-1} = 0$, $d_{-1} = 0$, and

$$\delta_k = \frac{1}{k+1}, \quad k = 0, 1, \ldots.$$

It is possible to show the convergence of $\Phi \hat{r}_k$ to the solution of the exploration-enhanced projected equation $\Phi r = \overline{\Pi} T^{(\lambda)}(\Phi r)$, assuming only that this equation has a unique solution (a contraction property is not neces-

sary since LSTD is not an iterative method, but rather approximates the projected equation by simulation).†

The exploration-enhanced LSPE(λ) iteration is given by

$$r_{k+1} = r_k - \gamma G_k \left(C_k^{(\lambda)} r_k - d_k^{(\lambda)} \right), \tag{6.128}$$

where G_k is a scaling matrix, γ is a positive stepsize, and $q_{k,t}$ are the modified temporal differences (6.124). Convergence of iteration (6.128) requires that G_k converges to a matrix G such that $I - \gamma G C^{(\lambda)}$ is a contraction. A favorable special case is the iterative regression method [cf. Eqs. (6.90) and (6.91)]

$$r_{k+1} = \left(C_k^{(\lambda)\prime} \Sigma_k^{-1} C_k^{(\lambda)} + \beta I \right)^{-1} \left(C_k^{(\lambda)\prime} \Sigma_k^{-1} d_k^{(\lambda)} + \beta r_k \right). \tag{6.129}$$

This method converges for any λ as it does not require that $T^{(\lambda)}$ is a contraction with respect to $\|\cdot\|_{\overline{\xi}}$. The corresponding LSPE(λ) method

$$r_{k+1} = r_k - \left(\frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t)\phi(i_t)' \right)^{-1} \left(C_k^{(\lambda)} r_k - d_k^{(\lambda)} \right)$$

is convergent only if $\overline{\Pi} T^{(\lambda)}$ is a contraction.

We finally note that an exploration-enhanced version of TD(λ) has been developed in [BeY09] (Section 5.3). It has the form

$$r_{k+1} = r_k - \gamma_k z_k \tilde{q}_{k,k},$$

where γ_k is a stepsize parameter and $\tilde{q}_{k,k}$ is the modified temporal difference

$$\tilde{q}_{k,k} = \phi(i_k)' r_k - \frac{p_{i_k i_{k+1}}}{\overline{p}_{i_k i_{k+1}}} \big(\alpha \phi(i_{k+1})' r_k + g(i_k, i_{k+1}) \big),$$

[cf. Eqs. (6.99) and (6.124)]. However, this method is guaranteed to converge to the solution of the exploration-enhanced projected equation $\Phi r = \overline{\Pi} T^{(\lambda)}(\Phi r)$ only if $\overline{\Pi} T^{(\lambda)}$ is a contraction. We next discuss conditions under which this is so.

† The analysis of the convergence $C_k^{(\lambda)} \to C^{(\lambda)}$ and $d_k^{(\lambda)} \to d^{(\lambda)}$ has been given in several sources under different assumptions: (a) In Nedić and Bertsekas [NeB03] for the case of policy evaluation in an α-discounted problem with no exploration ($\overline{P} = P$). (b) In Bertsekas and Yu [BeY09], assuming that $\alpha\lambda \max_{(i,j)}(p_{ij}/\overline{p}_{ij}) < 1$ (where we adopt the convention $0/0 = 0$), in which case the eligibility vectors z_k of Eq. (6.127) are generated by a contractive process and remain bounded. This covers the common case $\overline{P} = \big(1-\epsilon\big)P + \epsilon Q$, where $\epsilon > 0$ is a constant. The essential restriction here is that λ should be no more than $(1-\epsilon)$. (c) In Yu [Yu10a,b], for all $\lambda \in [0,1]$ and no other restrictions, in which case the eligibility vectors z_k typically become unbounded as k increases when $\lambda \max_{(i,j)}(a_{ij}/\ell_{ij}) > 1$. Mathematically, this is the most challenging analysis, and involves interesting stochastic phenomena.

Contraction Properties of Exploration-Enhanced Methods

We now consider the question whether $\Pi T^{(\lambda)}$ is a contraction. This is important for the corresponding LSPE(λ) and TD(λ)-type methods, which are valid only if $\Pi T^{(\lambda)}$ is a contraction, as mentioned earlier. Generally, $\Pi T^{(\lambda)}$ may not be a contraction. The key difficulty here is a potential norm mismatch: even if $T^{(\lambda)}$ is a contraction with respect to some norm, Π may not be nonexpansive with respect to the same norm.

We recall the definition

$$\overline{P} = (I - B)P + BQ,$$

[cf. Eq. (6.116)], where B is a diagonal matrix with the exploration probabilities $\beta_i \in [0, 1]$ on the diagonal, and Q is another transition probability matrix. The following proposition quantifies the restrictions on the size of the exploration probabilities in order to avoid the difficulty just described. Since Π is nonexpansive with respect to $\|\cdot\|_{\overline{\xi}}$, the proof is based on finding values of β_i for which $T^{(\lambda)}$ is a contraction with respect to $\|\cdot\|_{\overline{\xi}}$. This is equivalent to showing that the corresponding induced norm of the matrix

$$\alpha P^{(\lambda)} = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t (\alpha P)^{t+1} \tag{6.130}$$

[cf. Eq. (6.85)] is less than 1.

Proposition 6.3.6: Assume that $\overline{P}$ is irreducible and $\overline{\xi}$ is its invariant distribution. Then $T^{(\lambda)}$ and $\Pi T^{(\lambda)}$ are contractions with respect to $\|\cdot\|_{\overline{\xi}}$ for all $\lambda \in [0, 1)$ provided $\overline{\alpha} < 1$, where

$$\overline{\alpha} = \frac{\alpha}{\sqrt{1 - \max_{i=1,\ldots,n} \beta_i}}.$$

The associated modulus of contraction is at most equal to

$$\frac{\overline{\alpha}(1 - \lambda)}{1 - \overline{\alpha}\lambda}.$$

Proof: For all $z \in \Re^n$ with $z \neq 0$, we have

$$\|\alpha P z\|_{\overline{\xi}}^2 = \sum_{i=1}^{n} \overline{\xi}_i \left(\sum_{j=1}^{n} \alpha p_{ij} z_j \right)^2$$

$$
\begin{aligned}
&= \alpha^2 \sum_{i=1}^{n} \overline{\xi}_i \left(\sum_{j=1}^{n} p_{ij} z_j \right)^2 \\
&\leq \alpha^2 \sum_{i=1}^{n} \overline{\xi}_i \sum_{j=1}^{n} p_{ij} z_j^2 \\
&\leq \alpha^2 \sum_{i=1}^{n} \overline{\xi}_i \sum_{j=1}^{n} \frac{\overline{p}_{ij}}{1-\beta_i} z_j^2 \\
&\leq \frac{\alpha^2}{1-\beta} \sum_{j=1}^{n} \sum_{i=1}^{n} \bar{\xi}_i \overline{p}_{ij} z_j^2 \\
&= \overline{\alpha}^2 \sum_{j=1}^{n} \bar{\xi}_j z_j^2 \\
&= \overline{\alpha}^2 \|z\|_{\bar{\xi}}^2,
\end{aligned}
$$

where the first inequality follows from the convexity of the quadratic function, the second inequality follows from the fact $(1-\beta_i)p_{ij} \leq \overline{p}_{ij}$, and the next to last equality follows from the property

$$\sum_{i=1}^{n} \bar{\xi}_i \overline{p}_{ij} = \overline{\xi}_j$$

of the invariant distribution. Thus, αP is a contraction with respect to $\|\cdot\|_{\overline{\xi}}$ with modulus at most $\overline{\alpha}$.

Next we note that if $\overline{\alpha} < 1$, the norm of the matrix $\alpha P^{(\lambda)}$ of Eq. (6.130) is bounded by

$$(1-\lambda) \sum_{t=0}^{\infty} \lambda^t \|\alpha P\|_{\bar{\xi}}^{t+1} \leq (1-\lambda) \sum_{t=0}^{\infty} \lambda^t \overline{\alpha}^{t+1} = \frac{\overline{\alpha}(1-\lambda)}{1-\overline{\alpha}\lambda} < 1, \qquad (6.131)$$

from which the result follows. **Q.E.D.**

The preceding proposition delineates a range of values for the exploration probabilities in order for $\overline{\Pi} T^{(\lambda)}$ to be a contraction: it is sufficient that

$$\beta_i < 1 - \alpha^2, \qquad i = 1, \ldots, n,$$

independent of the value of λ. We next consider the effect of λ on the range of allowable exploration probabilities. While it seems difficult to fully quantify this effect, it appears that values of λ close to 1 tend to enlarge the range. In fact, $T^{(\lambda)}$ is a contraction with respect to any norm $\|\cdot\|_{\overline{\xi}}$ and consequently for any value of the exploration probabilities β_i, provided λ is sufficiently close to 1. This is shown in the following proposition.

Proposition 6.3.7: Given any exploration probabilities from the range $[0,1]$ such that $\overline{P}$ is irreducible, there exists $\overline{\lambda} \in [0,1)$ such that $T^{(\lambda)}$ and $\overline{\Pi}T^{(\lambda)}$ are contractions with respect to $\|\cdot\|_{\overline{\xi}}$ for all $\lambda \in [\overline{\lambda}, 1)$.

Proof: By Prop. 6.3.6, there exists $\overline{\xi}$ such that $\|\alpha P\|_{\overline{\xi}} < 1$. From Eq. (6.131) it follows that $\lim_{\lambda \to 1} \|\alpha P^{(\lambda)}\|_{\overline{\xi}} = 0$ and hence $\lim_{\lambda \to 1} \alpha P^{(\lambda)} = 0$. It follows that given any norm $\|\cdot\|$, $\alpha P^{(\lambda)}$ is a contraction with respect to that norm for λ sufficiently close to 1. In particular, this is true for any norm $\|\cdot\|_{\overline{\xi}}$, where $\overline{\xi}$ is the invariant distribution of an irreducible $\overline{P}$ that is generated with any exploration probabilities from the range $[0,1]$. **Q.E.D.**

We finally note that assuming $\overline{\Pi}T^{(\lambda)}$ is a contraction with modulus

$$\overline{\alpha}_\lambda = \frac{\overline{\alpha}(1-\lambda)}{1-\overline{\alpha}\lambda},$$

as per Prop. 6.3.6, we have the error bound

$$\|J_\mu - \Phi \overline{r}_\lambda\|_{\overline{\xi}} \leq \frac{1}{\sqrt{1-\overline{\alpha}_\lambda^2}} \|J_\mu - \overline{\Pi} J_\mu\|_{\overline{\xi}},$$

where $\Phi \overline{r}_\lambda$ is the fixed point of $\overline{\Pi}T^{(\lambda)}$. The proof is nearly identical to the one of Prop. 6.3.5.

6.3.8 Policy Oscillations – Chattering

We will now describe a generic mechanism that tends to cause policy oscillations in approximate policy iteration. To this end, we introduce the so called *greedy partition*. For a given approximation architecture $\tilde{J}(\cdot, r)$, this is a partition of the space $\Re^s$ of parameter vectors r into subsets R_μ, each subset corresponding to a stationary policy μ, and defined by

$$R_\mu = \big\{ r \mid T_\mu(\Phi r) = T(\Phi r) \big\}$$

or equivalently

$$R_\mu = \left\{ r \;\middle|\; \mu(i) = \arg \min_{u \in U(i)} \sum_{j=1}^n p_{ij}(u) \big(g(i,u,j) + \alpha \tilde{J}(j,r) \big),\ i = 1, \ldots, n \right\}.$$

Thus, R_μ is the set of parameter vectors r for which μ is greedy with respect to $\tilde{J}(\cdot, r)$.

We first consider the nonoptimistic version of approximate policy iteration. For simplicity, let us assume that we use a policy evaluation method (e.g., a projected equation or other method) that for each given μ produces a unique parameter vector denoted r_μ. Nonoptimistic policy iteration starts with a parameter vector r_0, which specifies μ^0 as a greedy policy with respect to $\tilde{J}(\cdot, r_0)$, and generates r_{μ^0} by using the given policy evaluation method. It then finds a policy μ^1 that is greedy with respect to $\tilde{J}(\cdot, r_{\mu^0})$, i.e., a μ^1 such that

$$r_{\mu^0} \in R_{\mu^1}.$$

It then repeats the process with μ^1 replacing μ^0. If some policy μ^k satisfying

$$r_{\mu^k} \in R_{\mu^k} \tag{6.132}$$

is encountered, the method keeps generating that policy. This is the necessary and sufficient condition for policy convergence in the nonoptimistic policy iteration method.

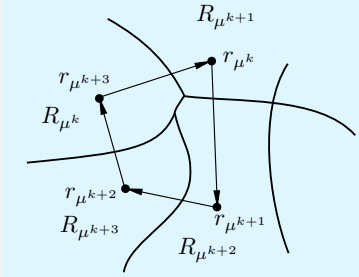

Figure 6.3.4 Greedy partition and cycle of policies generated by nonoptimistic policy iteration with cost function approximation. In particular, μ yields $\overline{\mu}$ by policy improvement if and only if $r_\mu \in R_{\overline{\mu}}$. In this figure, the method cycles between four policies and the corresponding four parameters r_{μ^k}, $r_{\mu^{k+1}}$, $r_{\mu^{k+2}}$, and $r_{\mu^{k+3}}$.

In the case of a lookup table representation where the parameter vectors r_μ are equal to the cost-to-go vector J_μ, the condition $r_{\mu^k} \in R_{\mu^k}$ is equivalent to $r_{\mu^k} = Tr_{\mu^k}$, and is satisfied if and only if μ^k is optimal. When there is cost function approximation, however, this condition need not be satisfied for any policy. Since there is a finite number of possible vectors r_μ, one generated from another in a deterministic way, the algorithm ends up repeating some cycle of policies $\mu^k, \mu^{k+1}, \ldots, \mu^{k+m}$ with

$$r_{\mu^k} \in R_{\mu^{k+1}},\ r_{\mu^{k+1}} \in R_{\mu^{k+2}}, \ldots, r_{\mu^{k+m-1}} \in R_{\mu^{k+m}},\ r_{\mu^{k+m}} \in R_{\mu^k}; \tag{6.133}$$

(see Fig. 6.3.4). Furthermore, there may be several different cycles, and the method may end up converging to any one of them. The actual cycle obtained depends on the initial policy μ^0. This is similar to gradient methods applied to minimization of functions with multiple local minima, where the limit of convergence depends on the starting point.

We now turn to examine policy oscillations in optimistic variants of policy evaluation methods with function approximation. Then the trajectory of the method is less predictable and depends on the fine details of the iterative policy evaluation method, such as the frequency of the policy updates and the stepsize used. Generally, given the current policy μ, optimistic policy iteration will move towards the corresponding "target" parameter r_μ, for as long as μ continues to be greedy with respect to the current cost-to-go approximation $\tilde{J}(\cdot, r)$, that is, for as long as the current parameter vector r belongs to the set R_μ. Once, however, the parameter r crosses into another set, say $R_{\overline{\mu}}$, the policy $\overline{\mu}$ becomes greedy, and r changes course and starts moving towards the new "target" $r_{\overline{\mu}}$. Thus, the "targets" r_μ of the method, and the corresponding policies μ and sets R_μ may keep changing, similar to nonoptimistic policy iteration. Simultaneously, the parameter vector r will move near the boundaries that separate the regions R_μ that the method visits, following reduced versions of the cycles that nonoptimistic policy iteration may follow (see Fig. 6.3.5). Furthermore, as Fig. 6.3.5 shows, if diminishing parameter changes are made between policy updates (such as for example when a diminishing stepsize is used by the policy evaluation method) and the method eventually cycles between several policies, the parameter vectors will tend to converge to the common boundary of the regions R_μ corresponding to these policies. This is the so-called *chattering* phenomenon for optimistic policy iteration, whereby there is simultaneously oscillation in policy space and convergence in parameter space.

An additional insight is that the choice of the iterative policy evaluation method (e.g., LSTD, LSPE, or TD for various values of λ) makes a difference in rate of convergence, but does not seem crucial for the quality of the final policy obtained (as long as the methods converge). Using a different value of λ changes the targets r_μ somewhat, but leaves the greedy partition unchanged. As a result, different methods "fish in the same waters" and tend to yield similar ultimate cycles of policies.

The following is an example of policy oscillations and chattering. Other examples are given in Section 6.4.2 of [BeT96] (Examples 6.9 and 6.10).

Example 6.3.2 (Policy Oscillation and Chattering)

Consider a discounted problem with two states, 1 and 2, illustrated in Fig. 6.3.6(a). There is a choice of control only at state 1, and there are two policies, denoted μ^* and μ. The optimal policy μ^*, when at state 1, stays at 1 with probability $p > 0$ and incurs a negative cost c. The other policy is μ and cycles between the two states with 0 cost. We consider linear approximation with a single feature $\phi(i)' = i$ for each of the states $i = 1, 2$, i.e.,

$$\Phi = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \qquad \tilde{J} = \Phi r = \begin{pmatrix} r \\ 2r \end{pmatrix}.$$

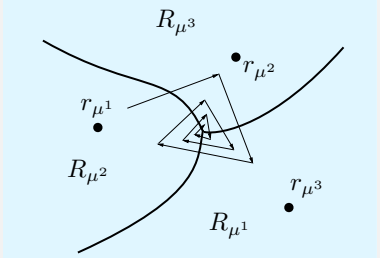

Figure 6.3.5 Illustration of a trajectory of optimistic policy iteration with cost function approximation. The algorithm settles into an oscillation between policies μ^1, μ^2, μ^3 with $r_{\mu^1} \in R_{\mu^2}$, $r_{\mu^2} \in R_{\mu^3}$, $r_{\mu^3} \in R_{\mu^1}$. The parameter vectors converge to the common boundary of these policies.

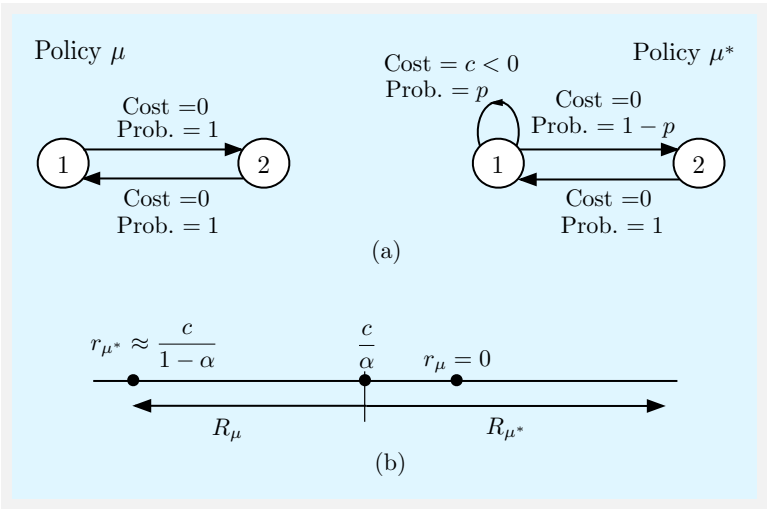

Figure 6.3.6 The problem of Example 6.3.2. (a) Costs and transition probabilities for the policies μ and μ^*. (b) The greedy partition and the solutions of the projected equations corresponding to μ and μ^*. Nonoptimistic policy iteration oscillates between r_μ and r_{μ^*}.

Let us construct the greedy partition. We have

$$\Phi = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \qquad \tilde{J} = \Phi r = \begin{pmatrix} r \\ 2r \end{pmatrix}.$$

We next calculate the points r_μ and r_{μ^*} that solve the projected equations

$$C_\mu r_\mu = d_\mu, \qquad C_{\mu^*} r_{\mu^*} = d_{\mu^*},$$

which correspond to μ and μ^*, respectively [cf. Eqs. (6.40), (6.41)]. We have

$$C_\mu = \Phi' \Xi_\mu (1 - \alpha P_\mu) \Phi = \begin{pmatrix} 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -\alpha \\ -a & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = 5 - 9\alpha,$$

$$d_\mu = \Phi' \Xi_\mu g_\mu = (\,1 \quad 2\,) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0,$$

so

$$r_\mu = 0.$$

Similarly, with some calculation,

$$\begin{aligned} C_{\mu^*} &= \Phi' \Xi_{\mu^*} (1 - \alpha P_{\mu^*}) \Phi \\ &= (\,1 \quad 2\,) \begin{pmatrix} \frac{1}{2-p} & 0 \\ 0 & \frac{1-p}{2-p} \end{pmatrix} \begin{pmatrix} 1 - \alpha p & -\alpha(1-p) \\ -a & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} \\ &= \frac{5 - 4p - \alpha(4 - 3p)}{2 - p}, \\ d_{\mu^*} &= \Phi' \Xi_{\mu^*} g_{\mu^*} = (\,1 \quad 2\,) \begin{pmatrix} \frac{1}{2-p} & 0 \\ 0 & \frac{1-p}{2-p} \end{pmatrix} \begin{pmatrix} c \\ 0 \end{pmatrix} = \frac{c}{2-p}, \end{aligned}$$

so

$$r_{\mu^*} = \frac{c}{5 - 4p - \alpha(4 - 3p)}.$$

We now note that since $c < 0$,

$$r_\mu = 0 \in R_{\mu^*},$$

while for $p \approx 1$ and $\alpha > 1 - \alpha$, we have

$$r_{\mu^*} \approx \frac{c}{1 - \alpha} \in R_\mu;$$

cf. Fig. 6.3.6(b). In this case, approximate policy iteration cycles between μ and μ^*. Optimistic policy iteration uses some algorithm that moves the current value r towards r_{μ^*} if $r \in R_{\mu^*}$, and towards r_μ if $r \in R_\mu$. Thus optimistic policy iteration starting from a point in R_μ moves towards r_{μ^*} and once it crosses the boundary point c/α of the greedy partition, it reverses course and moves towards r_μ. If the method makes small incremental changes in r before checking whether to change the current policy, it will incur a small oscillation around c/α. If the incremental changes in r are diminishing, the method will converge to c/α. Yet c/α does not correspond to any one of the two policies and has no meaning as a desirable parameter value.

Notice that it is hard to predict when an oscillation will occur and what kind of oscillation it will be. For example if $c > 0$, we have

$$r_\mu = 0 \in R_\mu,$$

while for $p \approx 1$ and $\alpha > 1 - \alpha$, we have

$$r_{\mu^*} \approx \frac{c}{1 - \alpha} \in R_{\mu^*}.$$

In this case approximate as well as optimistic policy iteration will converge to μ (or μ^*) if started with r in R_μ (or R_{μ^*}, respectively).

When chattering occurs, the limit of optimistic policy iteration tends to be on a common boundary of several subsets of the greedy partition and *may not meaningfully represent a cost approximation of any of the corresponding policies*, as illustrated by the preceding example. Thus, the limit to which the method converges cannot always be used to construct an approximation of the cost-to-go of any policy or the optimal cost-to-go. As a result, at the end of optimistic policy iteration and in contrast with the nonoptimistic version, one must go back and perform a screening process; that is, evaluate by simulation the many policies generated by the method starting from the initial conditions of interest and select the most promising one. This is a disadvantage of optimistic policy iteration that may nullify whatever practical rate of convergence advantages it may have over its nonoptimistic counterpart.

We note, however, that computational experience indicates that for many problems, the cost functions of the different policies involved in chattering may not be "too different." Indeed, suppose that we have convergence to a parameter vector $\overline{r}$ and that there is a steady-state policy oscillation involving a collection of policies $\mathcal{M}$. Then, all the policies in $\mathcal{M}$ are greedy with respect to $\tilde{J}(\cdot,\overline{r})$, which implies that there is a subset of states i such that there are at least two different controls $\mu_1(i)$ and $\mu_2(i)$ satisfying

$$\begin{aligned}\min_{u\in U(i)} \sum_j p_{ij}(u)\big(g(i,u,j)+\alpha\tilde{J}(j,\overline{r})\big) \\ = \sum_j p_{ij}\big(\mu_1(i)\big)\Big(g\big(i,\mu_1(i),j\big)+\alpha\tilde{J}(j,\overline{r})\Big) \\ = \sum_j p_{ij}\big(\mu_2(i)\big)\Big(g\big(i,\mu_2(i),j\big)+\alpha\tilde{J}(j,\overline{r})\Big).\end{aligned} \tag{6.134}$$

Each equation of this type can be viewed as a constraining relation on the parameter vector $\overline{r}$. Thus, excluding singular situations, there will be at most s relations of the form (6.134) holding, where s is the dimension of $\overline{r}$. This implies that there will be at most s "ambiguous" states where more than one control is greedy with respect to $\tilde{J}(\cdot,\overline{r})$ (in Example 6.3.6, state 1 is "ambiguous").

Now assume that we have a problem where the total number of states is much larger than s, and in addition there are no "critical" states; that is, the cost consequences of changing a policy in just a small number of states (say, of the order of s) is relatively small. It then follows that all policies in the set $\mathcal{M}$ involved in chattering have roughly the same cost. Furthermore, for the methods of this section, one may argue that the cost approximation $\tilde{J}(\cdot,\overline{r})$ is close to the cost approximation $\tilde{J}(\cdot,r_\mu)$ that would be generated for any of the policies $\mu\in\mathcal{M}$. Note, however, that the assumption of "no critical states," aside from the fact that it may not be easily quantifiable, it will not be true for many problems.

While the preceding argument may explain some of the observed empirical behavior, an important concern remains: even if all policies involved in chattering have roughly the same cost, it is still possible that none of them is particularly good; the policy iteration process may be just cycling in a "bad" part of the greedy partition. An interesting case in point is the game of tetris, which has been used as a testbed for approximate DP methods [Van95], [TsV96], [BeI96], [Kak02], [FaV06], [SzL06], [DFM09]. Using a set of 22 features and approximate policy iteration with policy evaluation based on the projected equation and the LSPE method [BeI96], an average score of a few thousands was achieved. Using the same features and a random search method in the space of weight vectors r, an average score of over 900,000 was achieved [ThS09]. This suggests that in the tetris problem, policy iteration using the projected equation may be seriously hampered by oscillations or by chattering between relatively poor policies, roughly similar to the attraction of gradient methods to poor local minima. The full ramifications of policy oscillation in practice are not fully understood at present, but it is clear that they give serious reason for concern, and future research may shed more light on this question. It should also be noted that local minima-type phenomena may be causing similar difficulties in other related approximate DP methodologies: approximate policy iteration with the Bellman error method (see Section 6.8.4), policy gradient methods (see Section 6.9), and approximate linear programming (the tetris problem, using the same 22 features, has been addressed by approximate linear programming [FaV06], [DFM09], and with a policy gradient method [Kak02], also with an achieved average score of a few thousands).

Conditions for Policy Convergence

The preceding discussion has illustrated the detrimental effects of policy oscillation in approximate policy iteration. Another reason why convergence of policies may be desirable has to do with error bounds. Generally, in approximate policy iteration, by Prop. 1.3.6, we have an error bound of the form

$$\limsup_{k\to\infty} \|J_{\mu^k} - J^*\|_\infty \leq \frac{2\alpha\delta}{(1-\alpha)^2},$$

where δ satisfies

$$\|J_k - J_{\mu^k}\|_\infty \leq \delta$$

for all generated policies μ^k and J_k is the approximate cost vector of μ^k that is used for policy improvement. However, when the policy sequence $\{\mu^k\}$ terminates with some $\overline{\mu}$ and the policy evaluation is accurate to within δ (in the sup-norm sense $\|\Phi r_\mu - J_\mu\|_\infty \leq \delta$, for all μ), then one can show the much sharper bound

$$\|J_{\overline{\mu}} - J^*\|_\infty \leq \frac{2\alpha\delta}{1-\alpha}. \tag{6.135}$$

For a proof, let $\overline{J}$ be the cost vector $\Phi r_{\overline{\mu}}$ obtained by policy evaluation of $\overline{\mu}$, and note that it satisfies $\|\overline{J} - J_{\overline{\mu}}\|_\infty \leq \delta$ (by our assumption on the accuracy of policy evaluation) and $T\overline{J} = T_{\overline{\mu}}\overline{J}$ (since μ^k terminates at $\overline{\mu}$). We write

$$TJ_{\overline{\mu}} \geq T(\overline{J}-\delta e) = T\overline{J}-\alpha\delta e = T_{\overline{\mu}}\overline{J}-\alpha\delta e \geq T_{\overline{\mu}}(J_{\overline{\mu}}-\delta e)-\alpha\delta e = T_{\overline{\mu}}J_{\overline{\mu}}-2\alpha\delta e,$$

and since $T_{\overline{\mu}}J_{\overline{\mu}} = J_{\overline{\mu}}$, we obtain $TJ_{\overline{\mu}} \geq J_{\overline{\mu}} - 2\alpha\delta e$. From this, by applying T to both sides, we obtain

$$T^2J_{\overline{\mu}} \geq TJ_{\overline{\mu}} - 2\alpha^2\delta e \geq J_{\overline{\mu}} - 2\alpha\delta(1+\alpha)e,$$

and by similar continued application of T to both sides,

$$J^* = \lim_{m\to\infty} T^m J_{\overline{\mu}} \geq J_{\overline{\mu}} - \frac{2\alpha\delta}{1-\alpha}e,$$

thereby showing the error bound (6.135).

In view of the preceding discussion, it is interesting to investigate conditions under which we have convergence of policies. From the mathematical point of view, it turns out that policy oscillation is caused by the lack of monotonicity of the projection operator ($J \leq J'$ does not imply that $\Pi J \leq \Pi J'$). Generally, monotonicity is an essential property for the convergence of policy iteration-type methods (see the proof of the following proposition). Changing the sampling policy at each iteration may also create problems as it changes the projection norm, and interferes with convergence proofs of policy iteration. With this in mind, we will replace Π with a constant operator W that has a monotonicity property. Moreover, it is simpler both conceptually and notationally to do this in a broader and more abstract setting that transcends discounted DP problems, thereby obtaining a more general approximate policy iteration algorithm.

To this end, consider a method involving a (possibly nonlinear) mapping $H_\mu : \Re^n \mapsto \Re^n$, parametrized by the policy μ, and the mapping $H : \Re^n \mapsto \Re^n$, defined by

$$HJ = \min_{\mu\in\mathcal{M}} H_\mu J, \tag{6.136}$$

where $\mathcal{M}$ is a finite subset of policies, and the minimization above is done separately for each component of $H_\mu J$, i.e.,

$$(HJ)(i) = \min_{\mu\in\mathcal{M}} (H_\mu J)(i), \qquad \forall\ i = 1,\ldots,n.$$

Abstract mappings of this type and their relation to DP have been studied in Denardo [Den67], Bertsekas [Ber77], and Bertsekas and Shreve [BeS78]. The discounted DP case corresponds to $H_\mu = T_\mu$ and $H = T$. Another

special case is a mapping H_μ that is similar to T_μ but arises in discounted semi-Markov problems. Nonlinear mappings H_μ also arise in the context of minimax DP problems and sequential games; see Shapley [Sha53], and [Den67], [Ber77], [BeS78].

We will construct a policy iteration method that aims to find an approximation to a fixed point of H, and evaluates a policy $\mu \in \mathcal{M}$ with a solution $\tilde{J}_\mu$ of the following fixed point equation in the vector J:

$$(WH_\mu)(J) = J, \tag{6.137}$$

where $W : \Re^n \mapsto \Re^n$ is a mapping (possibly nonlinear, but independent of μ). Policy evaluation by solving the projected equation corresponds to $W = \Pi$. Rather than specify properties of H_μ under which H has a unique fixed point (as in [Den67], [Ber77], and [BeS78]), it is simpler for our purposes to introduce corresponding assumptions on the mappings W and WH_μ. In particular, we assume the following:

(a) For each J, the minimum in Eq. (6.136) is attained, in the sense that there exists $\overline{\mu} \in \mathcal{M}$ such that $HJ = H_{\overline{\mu}}J$.

(b) For each $\mu \in \mathcal{M}$, the mappings W and WH_μ are monotone in the sense that

$$WJ \leq W\overline{J}, \quad (WH_\mu)(J) \leq (WH_\mu)(\overline{J}), \qquad \forall\ J, \bar{J} \in \Re^n \text{ with } J \leq \bar{J}. \tag{6.138}$$

(c) For each μ, the solution $\tilde{J}_\mu$ of Eq. (6.137) is unique, and for all J such that $(WH_\mu)(J) \leq J$, we have

$$\tilde{J}_\mu = \lim_{k\to\infty} (WH_\mu)^k(J).$$

Based on condition (a), we introduce a policy improvement operation that is similar to the case where $H_\mu = T_\mu$, i.e., the "improved" policy $\overline{\mu}$ satisfies $H_{\overline{\mu}}\tilde{J}_\mu = H\tilde{J}_\mu$. Note that condition (c) is satisfied if WH_μ is a contraction, while condition (b) is satisfied if W is a matrix with nonnegative components and H_μ is monotone for all μ.

Proposition 6.3.8: Let the preceding conditions (a)-(c) hold. Consider the policy iteration method that uses the fixed point $\tilde{J}_\mu$ of the mapping WH_μ for evaluation of the policy μ [cf. Eq. (6.137)], and the equation $H_{\overline{\mu}}\tilde{J}_\mu = H\tilde{J}_\mu$ for policy improvement. Assume that the method is initiated with some policy in $\mathcal{M}$, and it is operated so that it terminates when a policy $\overline{\mu}$ is obtained such that $H_{\overline{\mu}}\tilde{J}_{\overline{\mu}} = H\tilde{J}_{\overline{\mu}}$. Then the method terminates in a finite number of iterations, and the vector $\tilde{J}_{\overline{\mu}}$ obtained upon termination is a fixed point of WH.

Proof: Similar to the standard proof of convergence of (exact) policy iteration, we use the policy improvement equation $H_{\overline{\mu}}\tilde{J}_\mu = H\tilde{J}_\mu$, the monotonicity of W, and the policy evaluation Eq. (6.137) to write

$$(WH_{\overline{\mu}})(\tilde{J}_\mu) = (WH)(\tilde{J}_\mu) \leq (WH_\mu)(\tilde{J}_\mu) = \tilde{J}_\mu.$$

By iterating with the monotone mapping $WH_{\overline{\mu}}$ and by using condition (c), we obtain

$$\tilde{J}_{\overline{\mu}} = \lim_{k\to\infty} (WH_{\overline{\mu}})^k(\tilde{J}_\mu) \leq \tilde{J}_\mu.$$

There are finitely many policies, so we must have $\tilde{J}_{\overline{\mu}} = \tilde{J}_\mu$ after a finite number of iterations, which using the policy improvement equation $H_{\overline{\mu}}\tilde{J}_\mu = H\tilde{J}_\mu$, implies that $H_{\overline{\mu}}\tilde{J}_{\overline{\mu}} = H\tilde{J}_{\overline{\mu}}$. Thus the algorithm terminates with $\overline{\mu}$, and since $\tilde{J}_{\overline{\mu}} = (WH_{\overline{\mu}})(\tilde{J}_{\overline{\mu}})$, it follows that $\tilde{J}_{\overline{\mu}}$ is a fixed point of WH. **Q.E.D.**

An important special case where Prop. 6.3.8 applies and policies converge is when $H_\mu = T_\mu$, $H = T$, W is linear of the form $W = \Phi D$, where Φ and D are $n \times s$ and $s \times n$ matrices, respectively, whose rows are probability distributions, and the policy evaluation uses the linear feature-based approximation $\tilde{J}_\mu = \Phi r_\mu$. This is the case of policy evaluation by aggregation, which will be discussed in Section 6.4. Then it can be seen that W is monotone and that WT_μ is a sup-norm contraction (since W is nonexpansive with respect to the sup-norm), so that conditions (a)-(c) are satisfied.

Policy convergence as per Prop. 6.3.8 is also attained in the more general case where $W = \Phi D$, with the matrix W having nonnegative components, and row sums that are less than or equal to 1, i.e.,

$$\sum_{m=1}^{s} \Phi_{im} D_{mj} \geq 0, \qquad \forall\ i, j = 1, \ldots, n, \tag{6.139}$$

$$\sum_{m=1}^{s} \Phi_{im} \sum_{j=1}^{n} D_{mj} \leq 1, \qquad \forall\ i = 1, \ldots, n. \tag{6.140}$$

If Φ and D have nonnegative components, Eq. (6.139) is automatically satisfied, while Eq. (6.140) is equivalent to the set of n linear inequalities

$$\phi(i)'\zeta \leq 1, \qquad \forall\ i = 1, \ldots, n, \tag{6.141}$$

where $\phi(i)'$ is the ith row of Φ, and $\zeta \in \Re^s$ is the column vector of row sums of D, i.e., the one that has components†

$$\zeta(m) = \sum_{j=1}^{n} D_{mj}, \qquad \forall\ m = 1, \ldots, s.$$

Even in this more general case, the policy evaluation Eq. (6.137) can be solved by using simulation and low order calculations (see Section 6.8).

A special case arises when through a reordering of indexes, the matrix D can be partitioned in the form $D = (\,\Delta \quad 0\,)$, where Δ is a positive definite diagonal matrix with diagonal elements δ_m, $m = 1, \ldots, s$, satisfying

$$\sum_{m=1}^{s} \Phi_{im}\delta_m \leq 1, \qquad \forall\ i = 1, \ldots, n.$$

An example of a structure of this type arises in coarse grid discretization/aggregation schemes (Section 6.4).

When the projected equation approach is used ($W = \Pi$ where Π is a projection matrix) and the mapping H_μ is monotone in the sense that $H_\mu J \leq H_\mu \overline{J}$ for all $J, \overline{J} \in \Re^n$ with $J \leq \overline{J}$ (as it typically is in DP models), then the monotonicity assumption (6.138) is satisfied if Π is independent of the policy μ and satisfies $\Pi J \geq 0$ for all J with $J \geq 0$. This is true in particular when both Φ and $(\Phi'\Xi\Phi)^{-1}$ have nonnegative components, in view of the projection formula $\Pi = \Phi(\Phi'\Xi\Phi)^{-1}\Phi'\Xi$. A special case is when Φ is nonnegative and has linearly independent columns that are orthogonal with respect to the inner product $< x_1, x_2 >= x_1'\Xi x_2$, in which case $\Phi'\Xi\Phi$ is positive definite and diagonal.

† A column of Φ that has both positive and negative components may be replaced with the two columns that contain its positive and the opposite of its negative components. This will create a new nonnegative matrix Φ with as many as twice the number of columns, and will also enlarge the approximation subspace S (leading to no worse approximation). Then the matrix D may be optimized subject to $D \geq 0$ and the constraints (6.141), with respect to some performance criterion. Given a choice of $\Phi \geq 0$, an interesting question is how to construct effective algorithms for parametric optimization of a nonnegative matrix $W = \Phi D$, subject to the constraints (6.139)-(6.140). One possibility is to use

$$D = \gamma\, M\Phi'\Xi, \qquad W = \gamma\, \Phi M\Phi'\Xi,$$

where M is a positive definite diagonal replacement/approximation of $(\Phi'\Xi\Phi)^{-1}$ in the projection formula $\Pi = \Phi(\Phi'\Xi\Phi)^{-1}\Phi'\Xi$, and $\gamma > 0$ is a scalar parameter that is adjusted to ensure that the condition (c) of Prop. 6.3.8 is satisfied. Note that $\Phi'\Xi\Phi$ may be easily computed by simulation, but since W should be independent of μ, the same should be true for Ξ.

An example of the latter case is *hard aggregation*, where the state space $\{1,\ldots,n\}$ is partitioned in s nonempty subsets $I_1,\ldots,I_s$ and (cf. Section 6.4, and Vol. I, Section 6.3.4):

(1) The ℓth column of Φ has components that are 1 or 0 depending on whether they correspond to an index in I_ℓ or not.

(2) The ℓth row of D is a probability distribution $(d_{\ell 1},\ldots,d_{\ell n})$ whose components are positive depending on whether they correspond to an index in I_ℓ or not, i.e., $\sum_{j=1}^{n} d_{\ell j}=1$, $d_{\ell j}>0$ if $j\in I_\ell$, and $d_{\ell j}=0$ if $j\notin I_\ell$.

With these definitions of Φ and D, it can be verified that W is given by the projection formula

$$W=\Phi D=\Pi=\Phi(\Phi'\Xi\Phi)^{-1}\Phi'\Xi,$$

where Ξ is the diagonal matrix with the nonzero components of D along the diagonal. In fact Π can be written in the explicit form

$$(\Pi J)(i)=\sum_{j\in I_\ell} d_{\ell j}J(j),\qquad \forall\ i\in I_\ell,\ \ell=1,\ldots,s.$$

Thus Φ and Π have nonnegative components and assuming that D (and hence Π) is held constant, policy iteration converges.

6.3.9 λ-Policy Iteration

In this section we return to the idea of optimistic policy iteration and we discuss an alternative method, which connects with TD and with the multistep λ-methods. We first consider the case of a lookup table representation, and we discuss later the case where we use cost function approximation.

We view optimistic policy iteration as a process that generates a sequence of cost vector-policy pairs $\big\{(J_k,\mu^k)\big\}$, starting with some (J_0,μ^0). At iteration k, we generate an "improved" policy μ^{k+1} satisfying

$$T_{\mu^{k+1}}J_k=TJ_k, \tag{6.142}$$

and we then compute J_{k+1} as an approximate evaluation of the cost vector $J_{\mu^{k+1}}$ of μ^{k+1}. In the optimistic policy iteration method that we have discussed so far, J_{k+1} is obtained by several, say m_k, value iterations using μ^{k+1}:

$$J_{k+1}=T^{m_k}_{\mu^{k+1}}J_k. \tag{6.143}$$

We now introduce another method whereby J_{k+1} is instead obtained by a *single* value iteration using the mapping $T^{(\lambda)}_{\mu^{k+1}}$:

$$J_{k+1}=T^{(\lambda)}_{\mu^{k+1}}J_k, \tag{6.144}$$

where for any μ and $\lambda \in (0,1)$,

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

This is the mapping encountered in Section 6.3.6 [cf. Eqs. (6.84)-(6.85)]:

$$T_\mu^{(\lambda)} J = g_\mu^{(\lambda)} + \alpha P_\mu^{(\lambda)} J, \tag{6.145}$$

where

$$P_\mu^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \alpha^\ell \lambda^\ell P_\mu^{\ell+1}, \qquad g_\mu^{(\lambda)} = \sum_{\ell=0}^{\infty} \alpha^\ell \lambda^\ell P_\mu^\ell g_\mu = (I - \alpha\lambda P_\mu)^{-1} g_\mu. \tag{6.146}$$

We call the method of Eqs. (6.142) and (6.144) λ-*policy iteration*, and we will show shortly that its properties are similar to the ones of the standard method of Eqs. (6.142), (6.143).

Indeed, both mappings $T_{\mu^{k+1}}^{m_k}$ and $T_{\mu^{k+1}}^{(\lambda)}$ appearing in Eqs. (6.143) and (6.144), involve multiple applications of the value iteration mapping $T_{\mu^{k+1}}$: a fixed number m_k in the former case (with $m_k = 1$ corresponding to value iteration and $m_k \to \infty$ corresponding to policy iteration), and an exponentially weighted number in the latter case (with $\lambda = 0$ corresponding to value iteration and $\lambda \to 1$ corresponding to policy iteration). Thus *optimistic policy iteration and* λ*-policy iteration are similar*: they just control the accuracy of the approximation $J_{k+1} \approx J_{\mu^{k+1}}$ by applying value iterations in different ways.

The following proposition provides some basic properties of λ-policy iteration.

Proposition 6.3.9: Given $\lambda \in [0,1)$, J_k, and μ^{k+1}, consider the mapping W_k defined by

$$W_k J = (1-\lambda) T_{\mu^{k+1}} J_k + \lambda T_{\mu^{k+1}} J. \tag{6.147}$$

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

(b) The vector J_{k+1} generated next by the λ-policy iteration method of Eqs. (6.142), (6.144) is the unique fixed point of W_k.

Proof: (a) For any two vectors J and $\overline{J}$, using the definition (6.147) of W_k, we have

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

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

(b) We have

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

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

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

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

From part (b) of the preceding proposition, we see that the equation defining J_{k+1} is

$$J_{k+1}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^{k+1}(i)\big)\Big(\big(g(i,\mu^{k+1}(i),j\big) + (1-\lambda)\alpha J_k(j) + \lambda\alpha J_{k+1}(j)\Big). \tag{6.148}$$

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

(a) *As Bellman's equation for an infinite-horizon $\lambda\alpha$-discounted MDP* where μ^{k+1} is the only policy, and the cost per stage is

$$g\big(i,\mu^{k+1}(i),j\big) + (1-\lambda)\alpha J_k(j).$$

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

Note that the two MDP described above are potentially much easier than the original, because they involve a smaller effective discount factor ($\lambda\alpha$ versus α). The two interpretations of λ-policy iteration in terms of these MDP provide options for approximate simulation-based solution using cost function approximation, which we discuss next. The approximate

solution can be obtained by using the projected equation approach of this section, or another methodology such as the aggregation approach of Section 6.4. Moreover the solution may itself be approximated with a finite number of value iterations, i.e., the algorithm

$$J_{k+1} = W_k^{m_k} J_k, \qquad T_{\mu^{k+1}} J_k = T J_k, \tag{6.149}$$

in place of Eqs. (6.142), (6.144), where W_k is the mapping (6.147) and $m_k > 1$ is an integer. These value iterations may converge fast because they involve the smaller effective discount factor $\lambda\alpha$.

Implementations of λ-Policy Iteration

We will now discuss three alternative simulation-based implementations with cost function approximation $J \approx \Phi r$ and projection. The first implementation is based on the formula $J_{k+1} = T^{(\lambda)}_{\mu^{k+1}} J_k$. This is just a single iteration of PVI(λ) for evaluating $J_{\mu^{k+1}}$, and can be approximated by a single iteration of LSPE(λ):

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

It can be implemented in the manner discussed in Section 6.3.6, with a simulation trajectory generated by using μ^{k+1}.

The second implementation is based on a property mentioned earlier: Eq. (6.148) is Bellman's equation for the policy μ^{k+1} in the context of an optimal stopping problem. Thus to compute a function approximation to J_{k+1}, we may find by simulation an approximate solution of this equation, by using a function approximation to J_k and the appropriate cost function approximation methods for stopping problems. We will discuss methods of this type in Section 6.6, including analogs of LSTD(λ), LSPE(λ), and TD(λ). We will see that these methods are often able to deal much more comfortably with the issue of exploration, and do not require elaborate modifications of the type discussed in Section 6.4.1, particularly they involve a relatively short trajectory from any initial state to the termination state, followed by restart from a randomly chosen state (see the comments at the end of Section 6.6). Here the termination probability at each state is $1 - \lambda$, so for λ not very close to 1, the simulation trajectories are short. When the details of this implementation are fleshed out it we obtain the exploration-enhanced version of LSPE(λ) described in Section 6.3.6 (see [Ber11b] for a detailed development).

The third implementation, suggested and tested by Thiery and Scherrer [ThS10a], is based on the fixed point property of J_{k+1} [cf. Prop. 6.3.9(b)], and uses the projected version of the equation $W_k J = (1 - \lambda) T_{\mu^{k+1}} J_k + \lambda T_{\mu^{k+1}} J$ [cf. Eq. (6.147)]

$$\Phi r = (1 - \lambda)\Pi T_{\mu^{k+1}}(\Phi r_k) + \lambda \Pi T_{\mu^{k+1}}(\Phi r), \tag{6.150}$$

or equivalently

$$\Phi r = \Pi \left(g_{\mu^{k+1}} + \alpha(1-\lambda) P_{\mu^{k+1}} \Phi r_k + \alpha\lambda P_{\mu^{k+1}} \Phi r \right).$$

It can be seen that this is a projected equation in r, similar to the one discussed in Section 6.3.1 [cf. Eq. (6.37)]. In particular, the solution r_{k+1} solves the orthogonality equation [cf. Eq. (6.39)]

$$Cr = d(k),$$

where

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

so that

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

In a simulation-based implementation, the matrix C and the vector $d(k)$ are approximated similar to LSTD(0). However, r_{k+1} as obtained by this method, aims to approximate r_0^*, the limit of TD(0), not r_λ^*, the limit of TD(λ). To see this, suppose that this iteration is repeated an infinite number of times so it converges to a limit r^*. Then from Eq. (6.150), we have

$$\Phi r^* = (1-\lambda)\Pi T_{\mu^{k+1}}(\Phi r^*) + \lambda \Pi T_{\mu^{k+1}}(\Phi r^*).$$

which shows that $\Phi r^* = \Pi T_{\mu^{k+1}}(\Phi r^*)$, so $r^* = r_0^*$. Indeed the approximation via projection in this implementation is somewhat inconsistent: it is designed so that Φr_{k+1} is an approximation to $T^{(\lambda)}_{\mu^{k+1}}(\Phi r_k)$ yet as $\lambda \to 1$, from Eq. (6.150) we see that $\Phi r_{k+1} \to r_0^*$, not r_λ^*. Thus it would appear that while this implementation deals well with the issue of exploration, it may not deal well with the issue of bias. For further discussion, we refer to Bertsekas [Ber11b].

Convergence and Convergence Rate of λ-Policy Iteration

The following proposition shows the validity of the λ-policy iteration method and provides its convergence rate for the case of a lookup table representation. A similar result holds for the optimistic version (6.149).

Proposition 6.3.10: (Convergence for Lookup Table Case) Assume that $\lambda \in [0,1)$, and let (J_k, μ^k) be the sequence generated by the λ-policy iteration algorithm of Eqs. (6.142), (6.144). Then J_k converges to J^*. Furthermore, for all k greater than some index $\overline{k}$, we have

$$\|J_{k+1} - J^*\| \leq \frac{\alpha(1-\lambda)}{1-\alpha\lambda} \|J_k - J^*\|, \tag{6.151}$$

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

Proof: Let us first assume that $TJ_0 \leq J_0$. We show by induction that for all k, we have

$$J^* \leq TJ_{k+1} \leq J_{k+1} \leq TJ_k \leq J_k. \tag{6.152}$$

To this end, we fix k and we assume that $TJ_k \leq J_k$. We will show that $J^* \leq TJ_{k+1} \leq J_{k+1} \leq TJ_k$, and then Eq. (6.152) will follow from the hypothesis $TJ_0 \leq J_0$.

Using the fact $T_{\mu^{k+1}}J_k = TJ_k$ and the definition of W_k [cf. Eq. (6.147)], we have

$$W_kJ_k = T_{\mu^{k+1}}J_k = TJ_k \leq J_k.$$

It follows from the monotonicity of $T_{\mu^{k+1}}$, which implies monotonicity of W_k, that for all positive integers ℓ, we have $W_k^{\ell+1}J_k \leq W_k^{\ell}J_k \leq TJ_k \leq J_k$, so by taking the limit as $\ell \to \infty$, we obtain

$$J_{k+1} \leq TJ_k \leq J_k. \tag{6.153}$$

From the definition of W_k, we have

$$\begin{aligned} W_kJ_{k+1} &= T_{\mu^{k+1}}J_k + \lambda(T_{\mu^{k+1}}J_{k+1} - T_{\mu^{k+1}}J_k) \\ &= T_{\mu^{k+1}}J_{k+1} + (1-\lambda)(T_{\mu^{k+1}}J_k - T_{\mu^{k+1}}J_{k+1}), \end{aligned}$$

Using the already shown relation $J_k - J_{k+1} \geq 0$ and the monotonicity of $T_{\mu^{k+1}}$, we obtain $T_{\mu^{k+1}}J_k - T_{\mu^{k+1}}J_{k+1} \geq 0$, so that

$$T_{\mu^{k+1}}J_{k+1} \leq W_kJ_{k+1}.$$

Since $W_kJ_{k+1} = J_{k+1}$, it follows that

$$TJ_{k+1} \leq T_{\mu^{k+1}}J_{k+1} \leq J_{k+1}. \tag{6.154}$$

Finally, the above relation and the monotonicity of $T_{\mu^{k+1}}$ imply that for all positive integers ℓ, we have $T^{\ell}_{\mu^{k+1}}J_{k+1} \leq T_{\mu^{k+1}}J_{k+1}$, so by taking the limit as $\ell \to \infty$, we obtain

$$J^* \leq J_{\mu^{k+1}} \leq T_{\mu^{k+1}}J_{k+1}. \tag{6.155}$$

From Eqs. (6.153)-(6.155), we see that the inductive proof of Eq. (6.152) is complete.

From Eq. (6.152), it follows that the sequence J_k converges to some limit $\hat{J}$ with $J^* \leq \hat{J}$. Using the definition (6.147) of W_k, and the facts $J_{k+1} = W_kJ_{k+1}$ and $T_{\mu^{k+1}}J_k = TJ_k$, we have

$$J_{k+1} = W_kJ_{k+1} = TJ_k + \lambda(T_{\mu^{k+1}}J_{k+1} - T_{\mu^{k+1}}J_k),$$

so by taking the limit as $k \to \infty$ and by using the fact $J_{k+1} - J_k \to 0$, we obtain $\hat{J} = T\hat{J}$. Thus $\hat{J}$ is a solution of Bellman's equation, and it follows that $\hat{J} = J^*$.

To show the result without the assumption $TJ_0 \leq J_0$, note that we can replace J_0 by a vector $\hat{J}_0 = J_0 + se$, where $e = (1, \ldots, 1)$ and s is a scalar that is sufficiently large so that we have $T\hat{J}_0 \leq \hat{J}_0$; it can be seen that for any scalar $s \geq (1-\alpha)^{-1} \max_i \big(TJ_0(i) - J_0(i)\big)$, the relation $T\hat{J}_0 \leq \hat{J}_0$ holds. Consider the λ-policy iteration algorithm started with $(\hat{J}_0, \mu^0)$, and let $(\hat{J}_k, \hat{\mu}^k)$ be the generated sequence. Then it can be verified by induction that for all k we have

$$\hat{J}_k - J_k = \left(\frac{\alpha(1-\lambda)}{1-\alpha\lambda}\right)^k s, \qquad \hat{\mu}^k = \mu^k.$$

Hence $\hat{J}_k - J_k \to 0$. Since we have already shown that $\hat{J}_k \to J^*$, it follows that $J_k \to J^*$ as well.

Since $J_k \to J^*$, it follows that for all k larger than some index $\overline{k}$, μ^{k+1} is an optimal policy, so that $T_{\mu^{k+1}} J^* = TJ^* = J^*$. By using this fact and Prop. 6.3.5, we obtain for all $k \geq \overline{k}$,

$$\|J_{k+1} - J^*\| = \big\|T^{(\lambda)}_{\mu^{k+1}} J_k - T^{(\lambda)}_{\mu^{k+1}} J^*\big\| \leq \frac{\alpha(1-\lambda)}{1-\alpha\lambda}\|J_k - J^*\|.$$

Q.E.D.

For the case of cost function approximation, we have the following error bound, which resembles the one for approximate policy iteration (Prop. 1.3.6 in Chapter 1).

Proposition 6.3.11: (Error Bound for Cost Function Approximation Case) Let $\{\lambda_\ell\}$ be a sequence of nonnegative scalars with $\sum_{\ell=0}^{\infty} \lambda_\ell = 1$. Consider an algorithm that obtains a sequence of cost vector-policy pairs $\big\{(J_k, \mu_k)\big\}$, starting with some (J_0, μ_0), as follows: at iteration k, it generates an improved policy μ_{k+1} satisfying

$$T_{\mu_{k+1}} J_k = TJ_k,$$

and then it computes J_{k+1} by some method that satisfies

$$\left\| J_{k+1} - \sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}_{\mu_{k+1}} J_k \right\|_\infty \leq \delta,$$

where δ is some scalar. Then we have

$$\limsup_{k\to\infty} \|J_{\mu_k} - J^*\|_\infty \leq \frac{2\alpha\delta}{(1-\alpha)^2}.$$

For the proof of the proposition, we refer to Thiery and Scherrer [ThS10b]. Note that the proposition applies to both the standard optimistic policy iteration method ($\lambda_\ell = 1$ for a single value of ℓ and $\lambda_\ell = 0$ for all other values), and the λ-policy iteration method [$\lambda_\ell = (1-\lambda)\lambda^\ell$].

6.3.10 A Synopsis

Several algorithms for approximate evaluation of the cost vector J_μ of a single stationary policy μ in finite-state discounted problems have been given so far, and we will now summarize the analysis. We will also explain what can go wrong when the assumptions of this analysis are violated. We have focused on two types of algorithms:

(1) Direct methods, such as the batch and incremental gradient methods of Section 6.2, including TD(1). These methods allow for a nonlinear approximation architecture, and for a lot of flexibility in the collection of the cost samples that are used in the least squares optimization. For example, in direct methods, issues of exploration do not interfere with issues of convergence. The drawbacks of direct methods are that they are not well-suited for problems with large variance of simulation "noise," and they can also be very slow when implemented using gradient-like methods. The former difficulty is in part due to the lack of the parameter λ, which is used in other methods to reduce the variance/noise in the parameter update formulas.

(2) Indirect methods that are based on solution of a projected version of Bellman's equation. These are simulation-based methods that include approximate matrix inversion methods such as LSTD(λ), and iterative methods such as LSPE(λ) and its scaled versions, and TD(λ) (Sections 6.3.1-6.3.6).

The salient characteristics of our analysis of indirect methods are:

(a) For a given choice of $\lambda \in [0,1)$, all indirect methods aim to compute r_λ^*, the unique solution of the projected Bellman equation $\Phi r = \Pi T^{(\lambda)}(\Phi r)$. This equation is linear, of the form $C^{(\lambda)} r = d^{(\lambda)}$, and expresses the orthogonality of the vector $\Phi r - T^{(\lambda)}(\Phi r)$ and the approximation subspace S.

(b) We may use simulation and low-order matrix-vector calculations to approximate $C^{(\lambda)}$ and $d^{(\lambda)}$ with a matrix $C_k^{(\lambda)}$ and vector $d_k^{(\lambda)}$, respectively. The simulation may be supplemented with exploration enhancement, which suitably changes the projection norm to ensure adequate weighting of all states in the cost approximation. This is important in the context of policy iteration, as discussed in Section 6.3.7.

(c) The approximations $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ can be used in both types of methods: matrix inversion and iterative. The principal example of a matrix inversion method is LSTD(λ), which simply computes the solution

$$\hat{r}_k = (C_k^{(\lambda)})^{-1} d_k^{(\lambda)} \tag{6.156}$$

of the simulation-based approximation $C_k^{(\lambda)} r = d_k^{(\lambda)}$ of the projected equation. Principal examples of iterative methods is LSPE(λ) and its scaled versions,

$$r_{k+1} = r_k - \gamma G_k \big(C_k^{(\lambda)} r_k - d_k^{(\lambda)}\big). \tag{6.157}$$

TD(λ) is another major iterative method. It differs in an important way from LSPE(λ), namely it uses single-sample approximations of $C^{(\lambda)}$ and $d^{(\lambda)}$, which are much less accurate than $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$, and as a result it requires a diminishing stepsize to deal with the associated noise. A key property for convergence to r_λ^* of TD(λ) and the unscaled form of LSPE(λ) (without exploration enhancement) is that $T^{(\lambda)}$ is a contraction with respect to the projection norm $\|\cdot\|_\xi$, which implies that $\Pi T^{(\lambda)}$ is also a contraction with respect to the same norm.

(d) LSTD(λ) and LSPE(λ) are connected through the regularized regression-based form (6.91), which aims to deal effectively with cases where $C_k^{(\lambda)}$ is nearly singular and/or involves large simulation error (see Section 6.3.4). This is the special case of the LSPE(λ) class of methods, corresponding to the special choice (6.90) of G_k. The LSTD(λ) method of Eq. (6.156), and the entire class of LSPE(λ)-type iterations (6.157) converge at the same asymptotic rate, in the sense that

$$\|\hat{r}_k - r_k\| << \|\hat{r}_k - r_\lambda^*\|$$

for large k. However, depending on the choice of G_k, the short-term behavior of the LSPE-type methods is more regular as it involves implicit regularization through dependence on the initial condition. This behavior may be an advantage in the policy iteration context where optimistic variants, involving more noisy iterations, are used.

(e) When the LSTD(λ) and LSPE(λ) methods are exploration-enhanced for the purpose of embedding within an approximate policy iteration framework, their convergence properties become more complicated: LSTD(λ) and the regularized regression-based version (6.91) of LSPE(λ) converge to the solution of the corresponding (exploration-enhanced) projected equation for an arbitrary amount of exploration, but TD(λ) and other special cases of LSPE(λ) do so only for a limited amount of exploration and/or for λ sufficiently close to 1, as discussed in Section 6.3.7. On the other hand there is a special least-squares

based exploration-enhanced version of LSPE(λ) that overcomes this difficulty (cf. Section 6.3.6).

(f) The limit r_λ^* depends on λ. The estimate of Prop. 6.3.5 indicates that the approximation error $\|J_\mu - \Phi r_\lambda^*\|_\xi$ increases as the distance $\|J_\mu - \Pi J_\mu\|_\xi$ from the subspace S becomes larger, and also increases as λ becomes smaller. Indeed, the error degradation may be very significant for small values of λ, as shown by an example in [Ber95] (also reproduced in Exercise 6.9), where TD(0) produces a very bad solution relative to ΠJ_μ, which is the limit of the solution Φr_λ^* produced by TD(λ) as $\lambda \to 1$. (This example involves a stochastic shortest path problem, but can be modified to illustrate the same conclusion for discounted problems.) Note, however, that in the context of approximate policy iteration, the correlation between approximation error in the cost of the current policy and the performance of the next policy is somewhat unclear in practice (for example adding a constant to the cost of the current policy at every state does not affect the result of the policy improvement step).

(g) As $\lambda \to 1$, the size of the approximation error $\|J_\mu - \Phi r_\lambda^*\|_\xi$ tends to diminish, but the methods become more vulnerable to simulation noise, and hence require more sampling for good performance. Indeed, the noise in a simulation sample of an ℓ-stages cost vector $T^\ell J$ tends to be larger as ℓ increases, and from the formula

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

it can be seen that the simulation samples of $T^{(\lambda)}(\Phi r_k)$, used by LSTD(λ) and LSPE(λ), tend to contain more noise as λ increases. This is consistent with practical experience, which indicates that the algorithms tend to be faster and more reliable in practice when λ takes smaller values (or at least when λ is not too close to 1). Generally, there is no rule of thumb for selecting λ, which is usually chosen with some trial and error.

(h) TD(λ) is much slower than LSTD(λ) and LSPE(λ) [unless the number of basis functions s is extremely large, in which case the overhead for the linear algebra calculations that are inherent in LSTD(λ) and LSPE(λ) becomes excessive]. This can be traced to TD(λ)'s use of single-sample approximations of $C^{(\lambda)}$ and $d^{(\lambda)}$, which are much less accurate than $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$.

The assumptions under which convergence of LSTD(λ), LSPE(λ), and TD(λ) is usually shown include:

(i) The use of a linear approximation architecture Φr, with Φ satisfying the rank Assumption 6.3.2.

(ii) The use for simulation purposes of a Markov chain that has a steady-state distribution vector with positive components, which defines the projection norm. This is typically the Markov chain associated with the policy being evaluated, or an exploration-enhanced variant.

(iii) The use for simulation purposes of a Markov chain that defines a projection norm with respect to which $\Pi T^{(\lambda)}$ is a contraction. This is important only for some of the methods: TD(λ) and scaled LSPE(λ) [except for the regularized regression-based version (6.91)].

(iv) The use of a diminishing stepsize in the case of TD(λ). For LSTD(λ), and LSPE(λ) and its regularized regression-based form (6.91), there is no stepsize choice, and in various cases of scaled versions of LSPE(λ) the required stepsize is constant.

(v) The use of a single policy, unchanged during the simulation; convergence does not extend to the case where T involves a minimization over multiple policies, or optimistic variants, where the policy used to generate the simulation data is changed after a few transitions.

Let us now discuss the above assumptions (i)-(v). Regarding (i), there are no convergence guarantees for methods that use nonlinear architectures. In particular, an example in [TsV97] (also replicated in [BeT96], Example 6.6) shows that TD(λ) may diverge if a nonlinear architecture is used. In the case where Φ does not have rank s, the mapping $\Pi T^{(\lambda)}$ will still be a contraction with respect to $\|\cdot\|_\xi$, so it has a unique fixed point. In this case, TD(λ) has been shown to converge to *some* vector $r^* \in \Re^s$. This vector is the orthogonal projection of the initial guess r_0 on the set of solutions of the projected Bellman equation, i.e., the set of all r such that Φr is the unique fixed point of $\Pi T^{(\lambda)}$; see [Ber09b], [Ber11a]. LSPE(λ) and its scaled variants can be shown to have a similar property.

Regarding (ii), if we use for simulation a Markov chain whose steady-state distribution exists but has some components that are 0, the corresponding states are transient, so they will not appear in the simulation after a finite number of transitions. Once this happens, the algorithms will operate as if the Markov chain consists of just the recurrent states, and convergence will not be affected. However, the transient states would be underrepresented in the cost approximation. A similar difficulty occurs if we use for simulation a Markov chain with multiple recurrent classes. Then the results of the algorithms would depend on the initial state of the simulated trajectory (more precisely on the recurrent class of this initial state). In particular, states from other recurrent classes, and transient states would be underrepresented in the cost approximation obtained.

Regarding (iii), an example of divergence of TD(0) where the underlying projection norm is such that ΠT is not a contraction is given in [BeT96] (Example 6.7). Exercise 6.4 gives a similar example. On the other hand, as noted earlier, $\Pi T^{(\lambda)}$ is a contraction for any Euclidean projection

norm, provided λ is sufficiently close to 1.

Regarding (iv), the method for stepsize choice is critical for TD(λ); both for convergence and for performance. This is a major drawback of TD(λ), which compounds its practical difficulty with slow convergence.

Regarding (v), once minimization over multiple policies is introduced [so T and $T^{(\lambda)}$ are nonlinear], or optimistic variants are used, the behavior of the methods becomes quite peculiar and unpredictable because $\Pi T^{(\lambda)}$ may not be a contraction.† For instance, there are examples where $\Pi T^{(\lambda)}$ has no fixed point, and examples where it has multiple fixed points; see [BeT96] (Example 6.9), and [DFV00]. Generally, the issues associated with policy oscillations, the chattering phenomenon, and the asymptotic behavior of nonoptimistic and optimistic approximate policy iteration, are not well understood. Figures 6.3.4 and 6.3.5 suggest their enormously complex nature: the points where subsets in the greedy partition join are potential "points of attraction" of the various algorithms.

On the other hand, even in the case where $T^{(\lambda)}$ is nonlinear, if $\Pi T^{(\lambda)}$ is a contraction, it has a unique fixed point, and the peculiarities associated with chattering do not arise. In this case the scaled PVI(λ) iteration [cf. Eq. (6.44)] takes the form

$$r_{k+1} = r_k - \gamma G \Phi' \Xi \big(\Phi r_k - T^{(\lambda)}(\Phi r_k) \big),$$

where G is a scaling matrix, and γ is a positive stepsize that is small enough to guarantee convergence. As discussed in [Ber09b], [Ber11a], this iteration converges to the unique fixed point of $\Pi T^{(\lambda)}$, provided the constant stepsize γ is sufficiently small. Note that there are limited classes of problems, involving multiple policies, where the mapping $\Pi T^{(\lambda)}$ is a contraction. An example, optimal stopping problems, is discussed in Sections 6.5.3 and 6.8.3. Finally, let us note that the LSTD(λ) method relies on the linearity of the mapping T, and it has no practical generalization for the case where T is nonlinear.

6.4 AGGREGATION METHODS

In this section we revisit the aggregation methodology discussed in Section 6.3.4 of Vol. I, viewing it now in the context of policy evaluation with cost function approximation for discounted DP.† The aggregation approach resembles in some ways the problem approximation approach discussed in

† Similar to Prop. 6.3.5, it can be shown that $T^{(\lambda)}$ is a sup-norm contraction with modulus that tends to 0 as $\lambda \to 1$. It follows that given any projection norm $\|\cdot\|$, $T^{(\lambda)}$ and $\Pi T^{(\lambda)}$ are contractions with respect to $\|\cdot\|$, provided λ is sufficiently close to 1.

† Aggregation may be used in conjunction with any Bellman equation associated with the given problem. For example, if the problem admits post-decision

Section 6.3.3 of Vol. I: the original problem is approximated with a related "aggregate" problem, which is then solved exactly to yield a cost-to-go approximation for the original problem. Still, in other ways the aggregation approach resembles the projected equation/subspace approximation approach, most importantly because it constructs cost approximations of the form Φr, i.e., linear combinations of basis functions. However, there are important differences: in aggregation methods there are no projections with respect to Euclidean norms, the simulations can be done more flexibly, and from a mathematical point of view, the underlying contractions are with respect to the sup-norm rather than a Euclidean norm.

To construct an aggregation framework, we introduce a finite set $\mathcal{A}$ of aggregate states, and we introduce two (somewhat arbitrary) choices of probabilities, which relate the original system states with the aggregate states:

(1) For each aggregate state x and original system state i, we specify the *disaggregation probability* d_{xi} [we have $\sum_{i=1}^{n} d_{xi} = 1$ for each $x \in \mathcal{A}$]. Roughly, d_{xi} may be interpreted as the "degree to which x is represented by i."

(2) For each aggregate state y and original system state j, we specify the *aggregation probability* ϕ_{jy} (we have $\sum_{y \in \mathcal{A}} \phi_{jy} = 1$ for each $j = 1, \ldots, n$). Roughly, ϕ_{jy} may be interpreted as the "degree of membership of j in the aggregate state y." The vectors $\{\phi_{jy} \mid j = 1, \ldots, n\}$ may also be viewed as basis functions that will be used to represent approximations of the cost vectors of the original problem.

Let us mention a few examples:

(a) In *hard and soft aggregation* (Examples 6.3.9 and 6.3.10 of Vol. I), we group the original system states into subsets, and we view each subset as an aggregate state. In hard aggregation each state belongs to one and only one subset, and the aggregation probabilities are

$$\phi_{jy} = 1 \quad \text{if system state } j \text{ belongs to aggregate state/subset } y.$$

One possibility is to choose the disaggregation probabilities as

$$d_{xi} = 1/n_x \quad \text{if system state } i \text{ belongs to aggregate state/subset } x,$$

where n_x is the number of states of x (this implicitly assumes that all states that belong to aggregate state/subset y are "equally representative"). In soft aggregation, we allow the aggregate states/subsets to

---

states (cf. Section 6.1.4), the aggregation may be done using the corresponding Bellman equation, with potentially significant simplifications resulting in the algorithms of this section.

overlap, with the aggregation probabilities ϕ_{jy} quantifying the "degree of membership" of j in the aggregate state/subset y. The selection of aggregate states in hard and soft aggregation is an important issue, which is not fully understood at present. However, in specific practical problems, based on intuition and problem-specific knowledge, there are usually evident choices, which may be fine-tuned by experimentation.

(b) In various *discretization schemes*, each original system state j is associated with a convex combination of aggregate states:

$$j \sim \sum_{y \in \mathcal{A}} \phi_{jy} y,$$

for some nonnegative weights ϕ_{jx}, whose sum is 1, and which are viewed as aggregation probabilities (this makes geometrical sense if both the original and the aggregate states are associated with points in a Euclidean space, as described in Example 6.3.13 of Vol. I).

(c) In *coarse grid schemes* (cf. Example 6.3.12 of Vol. I and the subsequent example in Section 6.4.1), a subset of representative states is chosen, each being an aggregate state. Thus, each aggregate state x is associated with a unique original state i_x, and we may use the disaggregation probabilities $d_{xi} = 1$ for $i = i_x$ and $d_{xi} = 0$ for $i \neq i_x$. The aggregation probabilities are chosen as in the preceding case (b).

The aggregation approach approximates cost vectors with Φr, where $r \in \Re^s$ is a weight vector to be determined, and Φ is the matrix whose jth row consists of the aggregation probabilities $\phi_{j1}, \ldots, \phi_{js}$. Thus aggregation involves an approximation architecture similar to the one of projected equation methods: it uses as features the aggregation probabilities. Conversely, starting from a set of s features for each state, we may construct a *feature-based hard aggregation* scheme by grouping together states with "similar features." In particular, we may use a more or less regular partition of the feature space, which induces a possibly irregular partition of the original state space into aggregate states (all states whose features fall in the same set of the feature partition form an aggregate state). This is a general approach for passing from a feature-based approximation of the cost vector to an aggregation-based approximation (see also [BeT96], Section 3.1.2). Unfortunately, in the resulting aggregation scheme the number of aggregate states may become very large.

The aggregation and disaggregation probabilities specify a dynamical system involving both aggregate and original system states (cf. Fig. 6.4.1). In this system:

(i) From aggregate state x, we generate original system state i according to d_{xi}.

(ii) We generate transitions from original system state i to original system state j according to $p_{ij}(u)$, with cost $g(i,u,j)$.

(iii) From original system state j, we generate aggregate state y according to ϕ_{jy}.

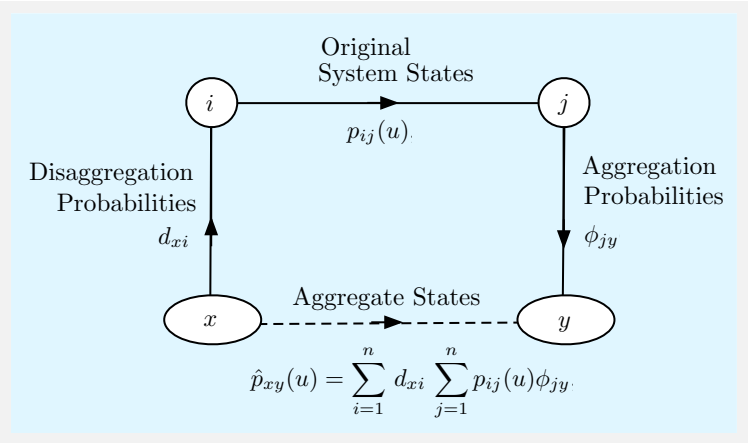

Figure 6.4.1 Illustration of the transition mechanism of a dynamical system involving both aggregate and original system states.

One may associate various DP problem formulations with this system, thereby obtaining two types of alternative cost approximations.

(a) In the first approximation, discussed in Section 6.4.1, the focus is on the aggregate states, the role of the original system states being to define the mechanisms of cost generation and probabilistic transition from one aggregate state to the next. This approximation may lead to small-sized aggregate problems that can be solved by ordinary value and policy iteration methods, even if the number of original system states is very large.

(b) In the second approximation, discussed in Section 6.4.2, the focus is on both the original system states and the aggregate states, which together are viewed as states of an enlarged system. Policy and value iteration algorithms are then defined for this enlarged system. For a large number of original system states, this approximation requires a simulation-based implementation.

6.4.1 Cost Approximation via the Aggregate Problem

Here we formulate an aggregate problem where the control is applied with knowledge of the aggregate state (rather than the original system state).

To this end, we assume that the control constraint set $U(i)$ is independent of the state i, and we denote it by U. Then, the transition probability from aggregate state x to aggregate state y under control u, and the corresponding expected transition cost, are given by (cf. Fig. 6.4.1)

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}, \quad \hat{g}(x,u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)g(i,u,j). \tag{6.158}$$

These transition probabilities and costs define an aggregate problem whose states are just the aggregate states.

The optimal cost function of the aggregate problem, denoted $\hat{J}$, is obtained as the unique solution of Bellman's equation

$$\hat{J}(x) = \min_{u \in U} \left[\hat{g}(x,u) + \alpha \sum_{y \in \mathcal{A}} p_{xy}(u)\hat{J}(y) \right], \qquad \forall\ x.$$

This equation has dimension equal to the number of aggregate states, and can be solved by any of the available value and policy iteration methods, including ones that involve simulation. Once $\hat{J}$ is obtained, the optimal cost function J^* of the original problem is approximated by $\tilde{J}$ given by

$$\tilde{J}(j) = \sum_{y \in \mathcal{A}} \phi_{jy}\hat{J}(y), \qquad \forall\ j,$$

which is used for one-step lookahead in the original system; i.e., a suboptimal policy $\overline{\mu}$ is obtained through the minimization

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

Note that for an original system state j, the approximation $\tilde{J}(j)$ is a convex combination of the costs $\hat{J}(y)$ of the aggregate states y for which $\phi_{jy} > 0$. In the case of hard aggregation, $\tilde{J}$ is piecewise constant: it assigns the same cost to all the states j that belong to the same aggregate state y (since $\phi_{jy} = 1$ if j belongs to y and $\phi_{jy} = 0$ otherwise).

The preceding scheme can also be applied to problems with infinite state space, and is well-suited for approximating the solution of partially observed Markov Decision problems (POMDP), which are defined over their belief space (space of probability distributions over their states, cf. Section 5.4.2 of Vol. I). By discretizing the belief space with a coarse grid, one obtains a *finite spaces* (aggregate) DP problem of perfect state information that can be solved with the methods of Chapter 1 (see [ZhL97], [ZhH01], [YuB04]). The following example illustrates the main ideas and shows that in the POMDP case, where the optimal cost function is a concave function over the simplex of beliefs (see Vol. I, Section 5.4.2), the approximation obtained is a lower bound of the optimal cost function.

Example 6.4.1 (Coarse Grid/POMDP Discretization and Lower Bound Approximations)

Consider an α-discounted DP problem with bounded cost per stage (cf. Section 1.2), where the state space is a convex subset C of a Euclidean space. We use z to denote the elements of this space, to distinguish them from x which now denotes aggregate states. Bellman's equation is $J = TJ$ with T defined by

$$(TJ)(z) = \min_{u \in U} \mathop{E}_{w} \Big\{ g(z,u,w) + \alpha J\big(f(z,u,w)\big) \Big\}, \qquad \forall\ z \in C.$$

Let J^* denote the optimal cost function. We select a finite subset/coarse grid of states $\{x_1, \ldots, x_m\} \in C$, whose convex hull is C. Thus each state $z \in C$ can be expressed as

$$z = \sum_{i=1}^{m} \phi_{z x_i} x_i,$$

where for each z, $\phi_{z x_i} \geq 0$, $i = 1, \ldots, m$, and $\sum_{i=1}^{m} \phi_{z x_i} = 1$. We view $\{x_1, \ldots, x_m\}$ as aggregate states with aggregation probabilities $\phi_{z x_i}$, $i = 1, \ldots, m$, for each $z \in C$. The disaggregation probabilities are $d_{x_k i} = 1$ for $i = x_k$ and $d_{x_k i} = 0$ for $i \neq x_k$, $k = 1, \ldots, m$. Consider the mapping $\hat{T}$ defined by

$$(\hat{T}J)(z) = \min_{u \in U} \mathop{E}_{w} \left\{ g(z,u,w) + \alpha \sum_{j=1}^{m} \phi_{f(z,u,w)\, x_j} J(x_j) \right\}, \qquad \forall\ z \in C,$$

where $\phi_{f(z,u,w)\, x_j}$ are the aggregation probabilities of the next state $f(z,u,w)$.

We note that $\hat{T}$ is a contraction mapping with respect to the sup-norm. Let $\hat{J}$ denotes its unique fixed point, so that we have

$$\hat{J}(x_i) = (\hat{T}\hat{J})(x_i), \qquad i = 1, \ldots, m.$$

This is Bellman's equation for an aggregated finite-state discounted DP problem whose states are $x_1, \ldots, x_m$, and can be solved by standard value and policy iteration methods that need not use simulation. We approximate the optimal cost function of the original problem by

$$\tilde{J}(z) = \sum_{i=1}^{m} \phi_{z x_i} \hat{J}(x_i), \qquad \forall\ z \in C.$$

Suppose now that J^* is a concave function over C (as in the POMDP case, where J^* is the limit of the finite horizon optimal cost functions that are concave, as shown in Vol. I, Section 5.4.2). Then for all (z,u,w), since $\phi_{f(z,u,w)\, x_j}$, $j = 1, \ldots, m$, are probabilities that add to 1, we have

$$J^*\big(f(z,u,w)\big) = J^* \left(\sum_{i=1}^{m} \phi_{f(z,u,w)\, x_i} x_i \right) \geq \sum_{i=1}^{m} \phi_{f(z,u,w)\, x_i} J^*(x_i);$$

this is a consequence of the definition of concavity and is also known as Jensen's inequality (see e.g., [Ber09a]). It then follows from the definitions of T and $\hat{T}$ that

$$J^*(z) = (TJ^*)(z) \geq (\hat{T}J^*)(z), \qquad \forall\ z \in C,$$

so by iterating, we see that

$$J^*(z) \geq \lim_{k\to\infty} (\hat{T}^k J^*)(z) = \hat{J}(z), \qquad \forall\ z \in C,$$

where the last equation follows because $\hat{T}$ is a contraction, and hence $\hat{T}^k J^*$ must converge to the unique fixed point $\hat{J}$ of $\hat{T}$. For $z = x_i$, we have in particular

$$J^*(x_i) \geq \hat{J}(x_i), \qquad \forall\ i = 1, \ldots, m,$$

from which we obtain for all $z \in C$,

$$J^*(z) = J^*\left(\sum_{i=1}^m \phi_{zx_i} x_i\right) \geq \sum_{i=1}^m \phi_{zx_i} J^*(x_i) \geq \sum_{i=1}^m \phi_{zx_i} \hat{J}(x_i) = \tilde{J}(z),$$

where the first inequality follows from the concavity of J^*. Thus the approximation $\tilde{J}(z)$ obtained from the aggregate system provides a lower bound to $J^*(z)$. Similarly, if J^* can be shown to be convex, the preceding argument can be modified to show that $\tilde{J}(z)$ is an upper bound to $J^*(z)$.

6.4.2 Cost Approximation via the Enlarged Problem

The approach of the preceding subsection calculates cost approximations assuming that policies assign controls to aggregate states, rather than to states of the original system. Thus, for example, in the case of hard aggregation, the calculations assume that the same control will be applied to every original system state within a given aggregate state. We will now discuss an alternative approach that is not subject to this limitation. Let us consider the system consisting of the original states and the aggregate states, with the transition probabilities and the stage costs described earlier (cf. Fig. 6.4.1). We introduce the vectors $\tilde{J}_0$, $\tilde{J}_1$, and R^* where:

$R^*(x)$ is the optimal cost-to-go from aggregate state x.

$\tilde{J}_0(i)$ is the optimal cost-to-go from original system state i that has just been generated from an aggregate state (left side of Fig. 6.4.1).

$\tilde{J}_1(j)$ is the optimal cost-to-go from original system state j that has just been generated from an original system state (right side of Fig. 6.4.1).

Note that because of the intermediate transitions to aggregate states, $\tilde{J}_0$ and $\tilde{J}_1$ are different.

These three vectors satisfy the following three Bellman's equations:

$$R^*(x) = \sum_{i=1}^{n} d_{xi}\tilde{J}_0(i), \qquad x \in \mathcal{A}, \tag{6.159}$$

$$\tilde{J}_0(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha\tilde{J}_1(j)\big), \qquad i = 1,\ldots,n, \tag{6.160}$$

$$\tilde{J}_1(j) = \sum_{y\in\mathcal{A}} \phi_{jy}R^*(y), \qquad j = 1,\ldots,n. \tag{6.161}$$

By combining these equations, we obtain an equation for R^*:

$$R^*(x) = (FR^*)(x), \qquad x \in \mathcal{A},$$

where F is the mapping defined by

$$(FR)(x) = \sum_{i=1}^{n} d_{xi} \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha\sum_{y\in\mathcal{A}}\phi_{jy}R(y)\right), \quad x \in \mathcal{A}. \tag{6.162}$$

It can be seen that F is a sup-norm contraction mapping and has R^* as its unique fixed point. This follows from standard contraction arguments (cf. Prop. 1.2.4) and the fact that d_{xi}, $p_{ij}(u)$, and ϕ_{jy} are all transition probabilities.†

Once R^* is found, the optimal-cost-to-go of the original problem may be approximated by $\tilde{J}_1 = \Phi R^*$, and a suboptimal policy may be found through the minimization (6.160) that defines $\tilde{J}_0$. Again, the optimal cost function approximation $\tilde{J}_1$ is a linear combination of the columns of Φ, which may be viewed as basis functions.

---

† A quick proof is to observe that F is the composition

$$F = DT\Phi,$$

where T is the usual DP mapping, and D and Φ are the matrices with rows the disaggregation and aggregation distributions, respectively. Since T is a contraction with respect to the sup-norm $\|\cdot\|_\infty$, and D and Φ are sup-norm nonexpansive in the sense

$$\|Dx\|_\infty \le \|x\|_\infty, \qquad \forall\ x \in \Re^n,$$

$$\|\Phi y\|_\infty \le \|y\|_\infty, \qquad \forall\ y \in \Re^s,$$

it follows that F is a sup-norm contraction.

Value and Policy Iteration

One may use value and policy iteration-type algorithms to find R^*. The value iteration algorithm simply generates successively $FR, F^2R, \ldots$, starting with some initial guess R. The policy iteration algorithm starts with a stationary policy μ^0 for the original problem, and given μ^k, it finds R_{μ^k} satisfying $R_{\mu^k} = F_{\mu^k} R_{\mu^k}$, where F_μ is the mapping defined by

$$(F_\mu R)(x) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big) \left(g\big(i, \mu(i), j\big) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy} R_\mu(y) \right), \quad x \in \mathcal{A}, \tag{6.163}$$

(this is the policy evaluation step). It then generates μ^{k+1} by

$$\mu^{k+1}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i, u, j) + \alpha \sum_{y \in \mathcal{A}} \phi_{jy} R_{\mu^k}(y) \right), \quad \forall\ i, \tag{6.164}$$

(this is the policy improvement step). Based on the discussion in Section 6.3.8 and Prop. 6.3.8, this policy iteration algorithm converges to the unique fixed point of F in a finite number of iterations. The key fact here is that F and F_μ are not only sup-norm contractions, but also have the monotonicity property of DP mappings (cf. Section 1.1.2 and Lemma 1.1.1), which was used in an essential way in the convergence proof of ordinary policy iteration (cf. Prop. 1.3.4).

As discussed in Section 6.3.8, when the policy sequence $\{\mu^k\}$ converges to some $\overline{\mu}$ as it does here, we have the error bound

$$\|J_{\overline{\mu}} - J^*\|_\infty \leq \frac{2\alpha\delta}{1-\alpha}, \tag{6.165}$$

where δ satisfies

$$\|J_k - J_{\mu^k}\|_\infty \leq \delta,$$

for all generated policies μ^k and J_k is the approximate cost vector of μ^k that is used for policy improvement (which is ΦR_{μ^k} in the case of aggregation). This is much sharper than the error bound

$$\limsup_{k \to \infty} \|J_{\mu^k} - J^*\|_\infty \leq \frac{2\alpha\delta}{(1-\alpha)^2},$$

of Prop. 1.3.6.

The preceding error bound improvement suggests that approximate policy iteration based on aggregation may hold some advantage in terms of approximation quality, relative to its projected equation-based counterpart. For a generalization of this idea, see Exercise 6.15. The price for this, however, is that the basis functions in the aggregation approach are restricted by the requirement that the rows of Φ must be probability distributions.

Simulation-Based Policy Iteration

The policy iteration method just described requires n-dimensional calculations, and is impractical when n is large. An alternative, which is consistent with the philosophy of this chapter, is to implement it by simulation, using a matrix inversion/LSTD-type method, as we now proceed to describe.

For a given policy μ, the aggregate version of Bellman's equation, $R = F_\mu R$, is linear of the form [cf. Eq. (6.163)]

$$R = DT_\mu(\Phi R),$$

where D and Φ are the matrices with rows the disaggregation and aggregation distributions, respectively, and T_μ is the DP mapping associated with μ, i.e.,

$$T_\mu J = g_\mu + \alpha P_\mu J,$$

with P_μ the transition probability matrix corresponding to μ, and g_μ is the vector whose ith component is

$$\sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)g\big(i,\mu(i),j\big).$$

We can thus write this equation as

$$ER = f,$$

where

$$E = I - \alpha DP\Phi, \qquad f = Dg, \tag{6.166}$$

in analogy with the corresponding matrix and vector for the projected equation [cf. Eq. (6.41)].

We may use low-dimensional simulation to approximate E and f based on a given number of samples, similar to Section 6.3.3 [cf. Eqs. (6.48) and (6.49)]. In particular, a sample sequence $\big\{(i_0,j_0),(i_1,j_1),\dots\big\}$ is obtained by first generating a sequence of states $\{i_0,i_1,\dots\}$ by sampling according to a distribution $\{\xi_i \mid i=1,\dots,n\}$ (with $\xi_i > 0$ for all i), and then by generating for each t the column index j_t using sampling according to the distribution $\{p_{i_t j} \mid j=1,\dots,n\}$. Given the first $k+1$ samples, we form the matrix $\hat{E}_k$ and vector $\hat{f}_k$ given by

$$\hat{E}_k = I - \frac{\alpha}{k+1}\sum_{t=0}^{k}\frac{1}{\xi_{i_t}}d(i_t)\phi(j_t)', \quad \hat{f}_k = \frac{1}{k+1}\sum_{t=0}^{k}\frac{1}{\xi_{i_t}}d(i_t)g\big(i_t,\mu(i_t),j_t\big), \tag{6.167}$$

where $d(i)$ is the ith column of D and $\phi(j)'$ is the jth row of Φ. The convergence $\hat{E}_k \to E$ and $\hat{f}_k \to f$ follows from the expressions

$$E = I - \alpha\sum_{i=1}^{n}\sum_{j=1}^{n}p_{ij}\big(\mu(i)\big)d(i)\phi(j)', \quad f = \sum_{i=1}^{n}\sum_{j=1}^{n}p_{ij}\big(\mu(i)\big)d(i)g\big(i,\mu(i),j\big),$$

the relation

$$\lim_{k\to\infty}\sum_{t=0}^{k}\frac{\delta(i_t=i,j_t=j)}{k+1}=\xi_i\,p_{ij},$$

and law of large numbers arguments (cf. Section 6.3.3).

It is important to note that the sampling probabilities ξ_i are restricted to be positive, but are otherwise arbitrary and need not depend on the current policy. Moreover, their choice does not affect the obtained approximate solution of the equation $ER=f$. Because of this possibility, the problem of exploration is less acute in the context of policy iteration when aggregation is used for policy evaluation. This is in contrast with the projected equation approach, where the choice of ξ_i affects the projection norm and the solution of the projected equation, as well as the contraction properties of the mapping ΠT.

Note also that instead of using the probabilities ξ_i to sample original system states, we may alternatively sample the aggregate states x according to a distribution $\{\zeta_x \mid x\in\mathcal{A}\}$, generate a sequence of aggregate states $\{x_0,x_1,\ldots\}$, and then generate a state sequence $\{i_0,i_1,\ldots\}$ using the disaggregation probabilities. In this case the equations (6.167) should be modified as follows:

$$\hat{E}_k=I-\frac{\alpha}{k+1}\sum_{t=0}^{k}\frac{1}{\zeta_{x_t}d_{x_ti_t}}d(i_t)\phi(j_t)',$$

$$\hat{f}_k=\frac{1}{k+1}\sum_{t=0}^{k}\frac{1}{\zeta_{x_t}d_{x_ti_t}}d(i_t)g\big(i_t,\mu(i_t),j_t\big).$$

The corresponding matrix inversion/LSTD-type method generates $\hat{R}_k=\hat{E}_k^{-1}\hat{f}_k$, and approximates the cost vector of μ by the vector $\Phi\hat{R}_k$:

$$\tilde{J}_\mu=\Phi\hat{R}_k.$$

There is also a regression-based version that is suitable for the case where $\hat{E}_k$ is nearly singular (cf. Section 6.3.4), as well as an iterative regression-based version of LSTD, which may be viewed as a special case of (scaled) LSPE. The latter method takes the form

$$\hat{R}_{k+1}=(\hat{E}_k'\Sigma_k^{-1}\hat{E}_k+\beta I)^{-1}(\hat{E}_k'\Sigma_k^{-1}\hat{f}_k+\beta\hat{R}_k),\tag{6.168}$$

where $\beta>0$ and Σ_k is a positive definite symmetric matrix [cf. Eq. (6.76)]. Note that contrary to the projected equation case, for a discount factor $\alpha\approx 1$, $\hat{E}_k$ will always be nearly singular [since $DP\Phi$ is a transition probability matrix, cf. Eq. (6.166)].

The nonoptimistic version of this aggregation-based policy iteration method does not exhibit the oscillatory behavior of the one based on the

projected equation approach (cf. Section 6.3.8): the generated policies converge and the limit policy satisfies the sharper error bound (6.165), as noted earlier. Moreover, optimistic versions of the method also do not exhibit the chattering phenomenon described in Section 6.3.8. This is similar to optimistic policy iteration for the case of a lookup table representation of the cost of the current policy: we are essentially dealing with a lookup table representation of the cost of the aggregate system of Fig. 6.4.1.

The preceding arguments indicate that aggregation-based policy iteration holds an advantage over its projected equation-based counterpart in terms of regularity of behavior, error guarantees, and exploration-related difficulties. Its limitation is that the basis functions in the aggregation approach are restricted by the requirement that the rows of Φ must be probability distributions. For example in the case of a single basis function ($s = 1$), there is only one possible choice for Φ in the aggregation context, namely the matrix whose single column is the unit vector.

Simulation-Based Value Iteration

The value iteration algorithm also admits a simulation-based implementation. It generates a sequence of aggregate states $\{x_0, x_1, \ldots\}$ by some probabilistic mechanism, which ensures that all aggregate states are generated infinitely often. Given each x_k, it independently generates an original system state i_k according to the probabilities $d_{x_k i}$, and updates $R(x_k)$ according to

$$R_{k+1}(x_k) = (1-\gamma_k)R_k(x_k) + \gamma_k \min_{u\in U(i)} \sum_{j=1}^{n} p_{i_k j}(u) \left(g(i_k, u, j) + \alpha \sum_{y\in\mathcal{A}} \phi_{jy} R_k(y) \right), \tag{6.169}$$

where γ_k is a diminishing positive stepsize, and leaves all the other components of R unchanged:

$$R_{k+1}(x) = R_k(x), \qquad \text{if } x \neq x_k.$$

This algorithm can be viewed as an asynchronous stochastic approximation version of value iteration. Its convergence mechanism and justification are very similar to the ones to be given for Q-learning in Section 6.5.1. It is often recommended to use a stepsize γ_k that depends on the state x_k being iterated on, such as for example $\gamma_k = 1/\big(1 + n(x_k)\big)$, where $n(x_k)$ is the number of times the state x_k has been generated in the simulation up to time k.

Multistep Aggregation

The aggregation methodology of this section can be generalized by considering a multistep aggregation-based dynamical system. This system,

illustrated in Fig. 6.4.2, is specified by disaggregation and aggregation probabilities as before, but involves $k > 1$ transitions between original system states in between transitions from and to aggregate states.

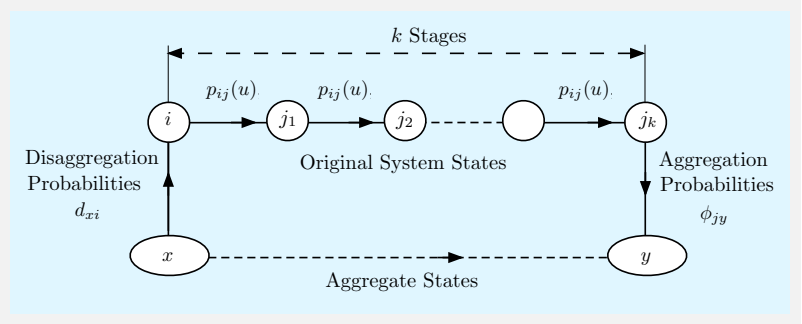

Figure 6.4.2 The transition mechanism for multistep aggregation. It is based on a dynamical system involving aggregate states, and k transitions between original system states in between transitions from and to aggregate states.

We introduce vectors $\tilde{J}_0, \tilde{J}_1, \ldots, \tilde{J}_k$, and R^* where:

$R^*(x)$ is the optimal cost-to-go from aggregate state x.

$\tilde{J}_0(i)$ is the optimal cost-to-go from original system state i that has just been generated from an aggregate state (left side of Fig. 6.4.2).

$\tilde{J}_1(j_1)$ is the optimal cost-to-go from original system state j that has just been generated from an original system state i.

$\tilde{J}_m(j_m)$, $m = 2, \ldots, k$, is the optimal cost-to-go from original system state j_m that has just been generated from an original system state j_{m-1}.

These vectors satisfy the following Bellman's equations:

$$R^*(x) = \sum_{i=1}^{n} d_{xi}\tilde{J}_0(i), \qquad x \in \mathcal{A},$$

$$\tilde{J}_0(i) = \min_{u \in U(i)} \sum_{j_1=1}^{n} p_{ij_1}(u)\big(g(i,u,j_1) + \alpha\tilde{J}_1(j_1)\big), \qquad i = 1, \ldots, n, \quad (6.170)$$

$$\tilde{J}_m(j_m) = \min_{u \in U(j_m)} \sum_{j_{m+1}=1}^{n} p_{j_m j_{m+1}}(u)\big(g(j_m,u,j_{m+1}) + \alpha\tilde{J}_{m+1}(j_{m+1})\big),$$
$$j_m = 1, \ldots, n, \ m = 1, \ldots, k-1, \quad (6.171)$$

$$\tilde{J}_k(j_k) = \sum_{y \in \mathcal{A}} \phi_{j_k y} R^*(y), \qquad j_k = 1, \ldots, n. \tag{6.172}$$

By combining these equations, we obtain an equation for R^*:

$$R^*(x) = (FR^*)(x), \qquad x \in \mathcal{A},$$

where F is the mapping defined by

$$FR = DT^k(\Phi R),$$

where T is the usual DP mapping of the problem. As earlier, it can be seen that F is a sup-norm contraction, but its contraction modulus is α^k rather than α.

There is a similar mapping corresponding to a fixed policy and it can be used to implement a policy iteration algorithm, which evaluates a policy through calculation of a corresponding vector R and then improves it. However, there is a major difference from the single-step aggregation case: a policy involves a set of k control functions $\{\mu_0, \ldots, \mu_{k-1}\}$, and while a known policy can be easily simulated, its improvement involves multistep lookahead using the minimizations of Eqs. (6.170)-(6.172), and may be costly. Thus multistep aggregation is a useful idea only for problems where the cost of this multistep lookahead minimization (for a single given starting state) is not prohibitive. On the other hand, note that from the theoretical point of view, a multistep scheme provides a means of better approximation of the true optimal cost vector J^*, independent of the use of a large number of aggregate states. This can be seen from Eqs. (6.170)-(6.172), which by classical value iteration convergence results, show that $\tilde{J}_0(i) \to J^*(i)$ as $k \to \infty$, regardless of the choice of aggregate states.

Asynchronous Distributed Aggregation

Let us now discuss the distributed solution of large-scale discounted DP problems using cost function approximation based on hard aggregation. We partition the original system states into aggregate states/subsets $x \in \mathcal{A} = \{x_1, \ldots, x_m\}$, and we envision a network of processors, each updating asynchronously a detailed/exact local cost function, defined on a single aggregate state/subset. Each processor also maintains an aggregate cost for its aggregate state, which is the weighted average detailed cost of the (original system) states in the processor's subset, weighted by the corresponding disaggregation probabilities. These aggregate costs are communicated between processors and are used to perform the local updates.

In a synchronous value iteration method of this type, each processor $a = 1, \ldots, m$, maintains/updates a (local) cost $J(i)$ for every original system state $i \in x_a$, and an aggregate cost

$$R(a) = \sum_{i \in x_a} d_{x_a i} J(i),$$

with $d_{x_a i}$ being the corresponding disaggregation probabilities. We generically denote by J and R the vectors with components $J(i)$, $i = 1, \ldots, n$, and $R(a)$, $a = 1, \ldots, m$, respectively. These components are updated according to

$$J_{k+1}(i) = \min_{u \in U(i)} H_a(i, u, J_k, R_k), \qquad \forall\ i \in x_a, \tag{6.173}$$

with

$$R_k(a) = \sum_{i \in x_a} d_{x_a i} J_k(i), \qquad \forall\ a = 1, \ldots, m, \tag{6.174}$$

where the mapping H_a is defined for all $a = 1, \ldots, m$, $i \in x_a$, $u \in U(i)$, and $J \in \Re^n$, $R \in \Re^m$, by

$$H_a(i, u, J, R) = \sum_{j=1}^{n} p_{ij}(u) g(i, u, j) + \alpha \sum_{j \in x_a} p_{ij}(u) J(j) + \alpha \sum_{j \notin x_a} p_{ij}(u) R\big(x(j)\big), \tag{6.175}$$

and where for each original system state j, we denote by $x(j)$ the subset to which i belongs [i.e., $j \in x(j)$]. Thus the iteration (6.173) is the same as ordinary value iteration, except that the aggregate costs $R\big(x(j)\big)$ are used for states j whose costs are updated by other processors.

It is possible to show that the iteration (6.173)-(6.174) involves a sup-norm contraction mapping of modulus α, so it converges to the unique solution of the system of equations in (J, R)

$$J(i) = \min_{u \in U(i)} H_a(i, u, J, R), \qquad R(a) = \sum_{i \in x_a} d_{x_a i} J(i), \qquad \forall\ i \in x_a,\ a = 1, \ldots, m; \tag{6.176}$$

This follows from the fact that $\{d_{x_a i} \mid i = 1, \ldots, n\}$ is a probability distribution. We may view the equations (6.176) as a set of Bellman equations for an "aggregate" DP problem, which similar to our earlier discussion, involves both the original and the aggregate system states. The difference from the Bellman equations (6.159)-(6.161) is that the mapping (6.175) involves $J(j)$ rather than $R\big(x(j)\big)$ for $j \in x_a$.

In the algorithm (6.173)-(6.174), all processors a must be updating their local costs $J(i)$ and aggregate costs $R(a)$ synchronously, and communicate the aggregate costs to the other processors before a new iteration may begin. This is often impractical and time-wasting. In a more practical asynchronous version of the method, the aggregate costs $R(a)$ may be outdated to account for communication "delays" between processors. Moreover, the costs $J(i)$ need not be updated for all i; it is sufficient that they are updated by each processor a only for a (possibly empty) subset of $I_{a,k} \in x_a$. In this case, the iteration (6.173)-(6.174) is modified to take the form

$$J_{k+1}(i) = \min_{u \in U(i)} H_a\big(i, u, J_k, R_{\tau_{1,k}}(1), \ldots, R_{\tau_{m,k}}(m)\big), \qquad \forall\ i \in I_{a,k}, \tag{6.177}$$

with $0 \leq \tau_{a,k} \leq k$ for $a = 1, \ldots, m$, and

$$R_\tau(a) = \sum_{i \in x_a} d_{x_a i} J_\tau(i), \qquad \forall\ a = 1, \ldots, m. \tag{6.178}$$

The differences $k - \tau_{a,k}$, $a = 1, \ldots, m$, in Eq. (6.177) may be viewed as "delays" between the current time k and the times $\tau_{a,k}$ when the corresponding aggregate costs were computed at other processors. For convergence, it is of course essential that every $i \in x_a$ belongs to $I_{a,k}$ for infinitely many k (so each cost component is updated infinitely often), and $\lim_{k\to\infty} \tau_{a,k} = \infty$ for all $a = 1, \ldots, m$ (so that processors eventually communicate more recently computed aggregate costs to other processors).

Asynchronous distributed DP methods of this type have been proposed and analyzed by the author in [Ber82]. Their convergence, based on the sup-norm contraction property of the mapping underlying Eq. (6.176), has been established in [Ber82] (see also [Ber83]). The monotonicity property is also sufficient to establish convergence, and this may be useful in the convergence analysis of related algorithms for other nondiscounted DP models. We also mention that asynchronous distributed policy iteration methods have been developed recently (see [BeY10b]).

6.5 Q-LEARNING

We now introduce another method for discounted problems, which is suitable for cases where there is no explicit model of the system and the cost structure (a model-free context). The method is related to value iteration and can be used directly in the case of multiple policies. Instead of approximating the cost function of a particular policy, it updates the Q-factors associated with an *optimal* policy, thereby avoiding the multiple policy evaluation steps of the policy iteration method.

In the discounted problem, the Q-factors are defined, for all pairs (i, u) with $u \in U(i)$, by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big).$$

Using Bellman's equation, we see that the Q-factors satisfy for all (i, u),

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{v \in U(j)} Q^*(j,v)\right), \tag{6.179}$$

and can be shown to be the unique solution of this set of equations. The proof is essentially the same as the proof of existence and uniqueness of the solution of Bellman's equation. In fact, by introducing a system whose

states are the original states $1, \ldots, n$, together with all the pairs (i, u), the above set of equations can be seen to be a special case of Bellman's equation (see Fig. 6.5.1). The Q-factors can be obtained by the value iteration $Q_{k+1} = FQ_k$, where F is the mapping defined by

$$(FQ)(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \min_{v \in U(j)} Q(j,v) \right), \qquad \forall\ (i,u). \tag{6.180}$$

Since F is a sup-norm contraction with modulus α (it corresponds to Bellman's equation for an α-discounted problem), this iteration converges from every starting point Q_0.

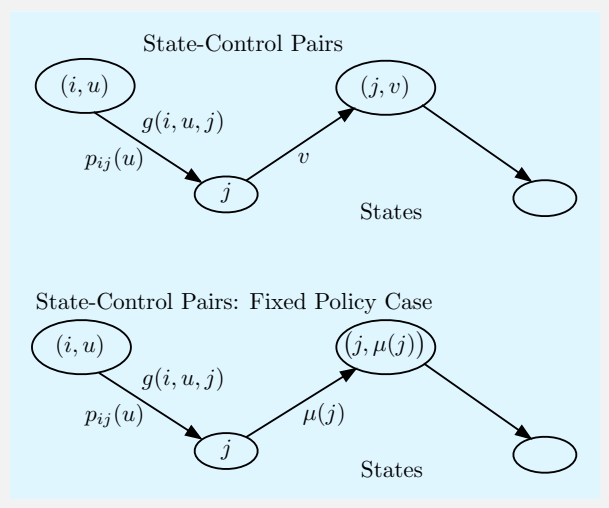

Figure 6.5.1 Modified problem where the state-control pairs (i, u) are viewed as additional states. The bottom figure corresponds to a fixed policy μ. The transitions from (i, u) to j are according to transition probabilities $p_{ij}(u)$ and incur a cost $g(i, u, j)$. Once the control v is chosen, the transitions from j to (j, v) occur with probability 1 and incur no cost.

The *Q-learning algorithm* is an approximate version of value iteration, whereby the expected value in Eq. (6.180) is suitably approximated by sampling and simulation. In particular, an infinitely long sequence of state-control pairs $\{(i_k, u_k)\}$ is generated according to some probabilistic mechanism. Given the pair (i_k, u_k), a state j_k is generated according to the probabilities $p_{i_k j}(u_k)$. Then the Q-factor of (i_k, u_k) is updated using a stepsize $\gamma_k > 0$ while all other Q-factors are left unchanged:

$$Q_{k+1}(i,u) = \begin{cases} (1-\gamma_k) Q_k(i,u) + \gamma_k (F_k Q_k)(i,u) & \text{if } (i,u) = (i_k, u_k), \\ Q_k(i,u) & \text{if } (i,u) = (i_k, u_k), \end{cases}$$

where

$$(F_kQ_k)(i_k,u_k) = g(i_k,u_k,j_k) + \alpha \min_{v\in U(j_k)} Q_k(j_k,v).$$

Equivalently,

$$Q_{k+1}(i,u) = (1-\gamma_k)Q_k(i,u) + \gamma_k(F_kQ_k)(i,u), \qquad \forall\ (i,u), \tag{6.181}$$

where

$$(F_kQ_k)(i,u) = \begin{cases} g(i_k,u_k,j_k) + \alpha \min_{v\in U(j_k)} Q_k(j_k,v) & \text{if } (i,u) = (i_k,u_k), \\ Q_k(i,u) & \text{if } (i,u) \neq (i_k,u_k). \end{cases} \tag{6.182}$$

To guarantee the convergence of the algorithm (6.181)-(6.182) to the optimal Q-factors, some conditions must be satisfied. Chief among them are that all state-control pairs (i,u) must be generated infinitely often within the infinitely long sequence $\{(i_k,u_k)\}$, and that the successor states j must be independently sampled at each occurrence of a given state-control pair. Furthermore, the stepsize γ_k should be diminishing to 0 at an appropriate rate, as we will discuss shortly.

Q-Learning and Aggregation

Let us also consider the use of Q-learning in conjunction with aggregation, involving a set $\mathcal{A}$ of aggregate states, disaggregation probabilities d_{ix} and aggregation probabilities ϕ_{jy}. The Q-factors $\hat{Q}(x,u)$, $x\in\mathcal{A}$, $u\in U$, of the aggregate problem of Section 6.4.1 are the unique solution of the Q-factor equation

$$\begin{aligned} \hat{Q}(x,u) &= \hat{g}(x,u) + \alpha \sum_{y\in\mathcal{A}} \hat{p}_{xy}(u) \min_{v\in U} \hat{Q}(y,v) \\ &= \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \sum_{y\in\mathcal{A}} \phi_{jy} \min_{v\in U} \hat{Q}(y,v) \right), \end{aligned} \tag{6.183}$$

[cf. Eq. (6.158)]. We may apply Q-learning to solve this equation. In particular, we generate an infinitely long sequence of pairs $\{(x_k,u_k)\} \subset \mathcal{A}\times U$ according to some probabilistic mechanism. For each (x_k,u_k), we generate an original system state i_k according to the disaggregation probabilities $d_{x_k i}$, and then a successor state j_k according to probabilities $p_{i_k j_k}(u_k)$. We finally generate an aggregate system state y_k using the aggregation probabilities $\phi_{j_k y}$. Then the Q-factor of (x_k,u_k) is updated using a stepsize $\gamma_k > 0$ while all other Q-factors are left unchanged [cf. Eqs. (6.181)-(6.182)]:

$$\hat{Q}_{k+1}(x,u) = (1-\gamma_k)\hat{Q}_k(x,u) + \gamma_k(F_k\hat{Q}_k)(x,u), \qquad \forall\ (x,u), \tag{6.184}$$

where the vector $F_k\hat{Q}_k$ is defined by

$$(F_k\hat{Q}_k)(x,u) = \begin{cases} g(i_k,u_k,j_k) + \alpha \min_{v\in U} \hat{Q}_k(y_k,v) & \text{if } (x,u) = (x_k,u_k), \\ \hat{Q}_k(x,u) & \text{if } (x,u) \neq (x_k,u_k). \end{cases}$$

Note that the probabilistic mechanism by which the pairs $\{(x_k,u_k)\}$ are generated is arbitrary, as long as all possible pairs are generated infinitely often. In practice, one may wish to use the aggregation and disaggregation probabilities, and the Markov chain transition probabilities in an effort to ensure that "important" state-control pairs are not underrepresented in the simulation.

After solving for the Q-factors $\hat{Q}$, the Q-factors of the original problem are approximated by

$$\tilde{Q}(j,v) = \sum_{y\in\mathcal{A}} \phi_{jy}\hat{Q}(y,v), \qquad j = 1,\ldots,n,\ v \in U. \tag{6.185}$$

We recognize this as an approximate representation $\tilde{Q}$ of the Q-factors of the original problem in terms of basis functions. There is a basis function for each aggregate state $y \in \mathcal{A}$ (the vector $\{\phi_{jy} \mid j = 1,\ldots,n\}$), and the corresponding coefficients that weigh the basis functions are the Q-factors of the aggregate problem $\hat{Q}(y,v)$, $y \in \mathcal{A}$, $v \in U$ (so we have in effect a lookup table representation with respect to v). The optimal cost-to-go function of the original problem is approximated by

$$\tilde{J}(j) = \min_{v\in U} \tilde{Q}(j,v), \qquad j = 1,\ldots,n,$$

and the corresponding one-step lookahead suboptimal policy is obtained as

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

Note that the preceding minimization requires knowledge of the transition probabilities $p_{ij}(u)$, which is unfortunate since a principal motivation of Q-learning is to deal with model-free situations where the transition probabilities are not explicitly known. The alternative is to obtain a suboptimal control at j by minimizing over $v \in U$ the Q-factor $\tilde{Q}(j,v)$ given by Eq. (6.185). This is less discriminating in the choice of control; for example in the case of hard aggregation, it applies the same control at all states j that belong to the same aggregate state y.

6.5.1 Convergence Properties of Q-Learning

We will explain the convergence properties of Q-learning, by viewing it as an asynchronous value iteration algorithm, where the expected value in the

definition (6.180) of the mapping F is approximated via a form of Monte Carlo averaging. In the process we will derive some variants of Q-learning that may offer computational advantages in some situations.†

In particular we will relate the Q-learning algorithm (6.181)-(6.182) to an (idealized) value iteration-type algorithm, which is defined by the same infinitely long sequence $\{(i_k, u_k)\}$, and is given by

$$Q_{k+1}(i,u) = \begin{cases} (FQ_k)(i_k,u_k) & \text{if } (i,u) = (i_k,u_k), \\ Q_k(i,u) & \text{if } (i,u) \neq (i_k,u_k), \end{cases} \tag{6.186}$$

where F is the mapping (6.180). Compared to the Q-learning algorithm (6.181)-(6.182), we note that this algorithm:

(a) Also updates at iteration k only the Q-factor corresponding to the pair (i_k, u_k), while it leaves all the other Q-factors are left unchanged.

(b) Involves the mapping F in place of F_k and a stepsize equal to 1 instead of γ_k.

We can view the algorithm (6.186) as a special case of an asynchronous value iteration algorithm of the type discussed in Section 1.3. Using the analysis of Gauss-Seidel value iteration and related methods given in that section, it can be shown that the algorithm (6.186) converges to the optimal Q-factor vector provided all state-control pairs (i, u) are generated infinitely often within the sequence $\{(i_k, u_k)\}$.†

---

† Much of the theory of Q-learning can be generalized to problems with post-decision states, where $p_{ij}(u)$ is of the form $q\big(f(i,u), j\big)$ (cf. Section 6.1.4). In particular, for such problems one may develop similar asynchronous simulation-based versions of value iteration for computing the optimal cost-to-go function V^* of the post-decision states $m = f(i,u)$: the mapping F of Eq. (6.180) is replaced by the mapping H given by

$$(HV)(m) = \sum_{j=1}^{n} q(m,j) \min_{u \in U(j)} \big[g(j,u) + \alpha V\big(f(j,u)\big) \big], \qquad \forall\ m,$$

[cf. Eq. (6.11)]. Q-learning corresponds to the special case where $f(i,u) = (i,u)$.

† Generally, iteration with a mapping that is either a contraction with respect to a weighted sup-norm, or has some monotonicity properties and a fixed point, converges when executed asynchronously (i.e., with different frequencies for different components, and with iterates that are not up-to-date). One or both of these properties are present in discounted and stochastic shortest path problems. As a result, there are strong asynchronous convergence guarantees for value iteration for such problems, as shown in [Ber82]. A general convergence theory of distributed asynchronous algorithms was developed in [Ber83] and has formed the basis for the convergence analysis of totally asynchronous algorithms in the book [BeT89].

Suppose now that we replace the expected value in the definition (6.180) of F, with a Monte Carlo estimate based on all the samples up to time k that involve (i_k, u_k). Letting n_k be the number of times the current state control pair (i_k, u_k) has been generated up to and including time k, and

$$T_k = \big\{t \mid (i_t, u_t) = (i_k, u_k),\ 0 \le t \le k\big\}$$

be the set of corresponding time indexes, we obtain the following algorithm:

$$Q_{k+1}(i,u) = (\tilde{F}_k Q_k)(i,u), \qquad \forall\ (i,u), \tag{6.187}$$

where $\tilde{F}_k Q_k$ is defined by

$$(\tilde{F}_k Q_k)(i_k,u_k) = \frac{1}{n_k}\sum_{t\in T_k}\left(g(i_t,u_t,j_t) + \alpha \min_{v\in U(j_t)} Q_k(j_t,v)\right), \tag{6.188}$$

$$(\tilde{F}_k Q_k)(i,u) = Q_k(i,u), \qquad \forall\ (i,u) \neq (i_k,u_k). \tag{6.189}$$

Comparing the preceding equations and Eq. (6.180), and using the law of large numbers, it is clear that for each (i,u), we have with probability 1

$$\lim_{k\to\infty,\ k\in T(i,u)} (\tilde{F}_k Q_k)(i,u) = (FQ_k)(i,u),$$

where $T(i,u) = \big\{k \mid (i_k,u_k) = (i,u)\big\}$. From this, the sup-norm contraction property of F, and the attendant asynchronous convergence properties of the value iteration algorithm $Q := FQ$, it can be shown that the algorithm (6.187)-(6.189) converges with probability 1 to the optimal Q-factors [assuming again that all state-control pairs (i,u) are generated infinitely often within the sequence $\{(i_k,u_k)\}$].

From the point of view of convergence rate, the algorithm (6.187)-(6.189) is quite satisfactory, but unfortunately it may have a significant drawback: it requires excessive overhead per iteration to calculate the Monte Carlo estimate $(\tilde{F}_k Q_k)(i_k,u_k)$ using Eq. (6.189). In particular, while the term $\frac{1}{n_k}\sum_{t\in T_k} g(i_t,u_t,j_t)$, in this equation can be recursively updated with minimal overhead, the term

$$\frac{1}{n_k}\sum_{t\in T_k} \min_{v\in U(j_t)} Q_k(j_t,v) \tag{6.190}$$

must be *completely recomputed* at each iteration k, using the current vector Q_k. This may be impractical, since the above summation may potentially involve a very large number of terms.†

Motivated by the preceding concern, let us modify the algorithm and replace the offending term (6.190) in Eq. (6.189) with

$$\frac{1}{n_k}\sum_{t\in T_k}\min_{v\in U(j_t)}Q_t(j_t,v), \tag{6.192}$$

which can be computed recursively, with minimal extra overhead. This is the algorithm (6.187), but with the Monte Carlo average $(\tilde{F}_kQ_k)(i_k,u_k)$ of Eq. (6.189) approximated by replacing the term (6.190) with the term (6.192), which depends on all the iterates Q_t, $t\in T_k$. This algorithm has the form

$$Q_{k+1}(i_k,u_k)=\frac{1}{n_k}\sum_{t\in T_k}\left(g(i_t,u_t,j_t)+\alpha\min_{v\in U(j_t)}Q_t(j_t,v)\right),\qquad\forall\ (i,u), \tag{6.193}$$

and

$$Q_{k+1}(i,u)=Q_k(i,u),\qquad\forall\ (i,u)\neq(i_k,u_k). \tag{6.194}$$

† We note a special type of problem where the overhead involved in updating the term (6.190) may be manageable. This is the case where for each pair (i,u) the set $S(i,u)$ of possible successor states j [the ones with $p_{ij}(u)>0$] has small cardinality. Then for each (i,u), we may maintain the numbers of times that each successor state $j\in S(i,u)$ has occurred up to time k, and use them to compute efficiently the troublesome term (6.190). In particular, we may implement the algorithm (6.187)-(6.189) as

$$Q_{k+1}(i_k,u_k)=\sum_{j\in S(i_k,u_k)}\frac{n_k(j)}{n_k}\left(g(i_k,u_k,j)+\alpha\min_{v\in U(j)}Q_k(j,v)\right), \tag{6.191}$$

where $n_k(j)$ is the number of times the transition (i_k,j), $j\in S(i_k,u_k)$, occurred at state i_k under u_k, in the simulation up to time k, i.e., $n_k(j)$ is the cardinality of the set

$$\{j_t=j\mid t\in T_k\},\qquad j\in S(i_k,u_k).$$

Note that this amounts to replacing the probabilities $p_{i_kj}(u_k)$ in the mapping (6.180) with their Monte Carlo estimates $\frac{n_k(j)}{n_k}$. While the minimization term in Eq. (6.191), $\min_{v\in U(j)}Q_k(j,v)$, has to be computed for all $j\in S(i_k,u_k)$ [rather than for just j_k as in the Q-learning algorithm algorithm (6.181)-(6.182)] the extra computation is not excessive if the cardinalities of $S(i_k,u_k)$ and $U(j)$, $j\in S(i_k,u_k)$, are small. This approach can be strengthened if some of the probabilities $p_{i_kj}(u_k)$ are known, in which case they can be used directly in Eq. (6.191). Generally, any estimate of $p_{i_kj}(u_k)$ can be used in place of $n_k(j)/n_k$, as long as the estimate converges to $p_{i_kj}(u_k)$ as $k\to\infty$.

We now show that this (approximate) value iteration algorithm is essentially the Q-learning algorithm (6.181)-(6.182).†

Indeed, let us observe that the iteration (6.193) can be written as

$$Q_{k+1}(i_k,u_k) = \frac{n_k-1}{n_k}Q_k(i_k,u_k)+\frac{1}{n_k}\left(g(i_k,u_k,j_k)+\alpha \min_{v\in U(j_k)} Q_k(j_k,v)\right),$$

or

$$Q_{k+1}(i_k,u_k) = \left(1-\frac{1}{n_k}\right)Q_k(i_k,u_k)+\frac{1}{n_k}(F_kQ_k)(i_k,u_k),$$

where $(F_kQ_k)(i_k,u_k)$ is given by the expression (6.182) used in the Q-learning algorithm. Thus the algorithm (6.193)-(6.194) is the Q-learning algorithm (6.181)-(6.182) with a stepsize $\gamma_k = 1/n_k$. It can be similarly shown that the algorithm (6.193)-(6.194), equipped with a stepsize parameter, is equivalent to the Q-learning algorithm with a different stepsize, say

$$\gamma_k = \frac{\gamma}{n_k},$$

where γ is a positive constant.

The preceding analysis provides a view of Q-learning as an approximation to asynchronous value iteration (updating one component at a time) that uses Monte Carlo sampling in place of the exact expected value in the mapping F of Eq. (6.180). It also justifies the use of a diminishing stepsize that goes to 0 at a rate proportional to $1/n_k$, where n_k is the number of times the pair (i_k,u_k) has been generated up to time k. However, it does not constitute a convergence proof because the Monte Carlo estimate used

† A potentially more effective algorithm is to introduce a window of size $m \geq 0$, and consider a more general scheme that calculates the last m terms of the sum in Eq. (6.190) exactly and the remaining terms according to the approximation (6.192). This algorithm, a variant of Q-learning, replaces the offending term (6.190) by

$$\frac{1}{n_k}\left(\sum_{t\in T_k,\, t\leq k-m} \min_{v\in U(j_t)} Q_{t+m}(j_t,v) + \sum_{t\in T_k,\, t>k-m} \min_{v\in U(j_t)} Q_k(j_t,v)\right), \quad (6.195)$$

which may also be updated recursively. The algorithm updates at time k the values of $\min_{v\in U(j_t)} Q(j_t,v)$ to $\min_{v\in U(j_t)} Q_k(j_t,v)$ for all $t \in T_k$ within the window $k-m \leq t \leq k$, and fixes them at the last updated value for t outside this window. For $m = 0$, it reduces to the algorithm (6.193)-(6.194). For moderate values of m it involves moderate additional overhead, and it is likely a more accurate approximation to the term (6.190) than the term (6.192) [$\min_{v\in U(j_t)} Q_{t+m}(j_t,v)$ presumably approximates better than $\min_{v\in U(j_t)} Q_t(j_t,v)$ the "correct" term $\min_{v\in U(j_t)} Q_k(j_t,v)$].

to approximate the expected value in the definition (6.180) of F is accurate only in the limit, if Q_k converges. We refer to Tsitsiklis [Tsi94] for a rigorous proof of convergence of Q-learning, which uses the theoretical machinery of stochastic approximation algorithms.

In practice, despite its theoretical convergence guaranties, Q-learning has some drawbacks, the most important of which is that the number of Q-factors/state-control pairs (i, u) may be excessive. To alleviate this difficulty, we may introduce a state aggregation scheme. Alternatively, we may introduce a linear approximation architecture for the Q-factors, similar to the policy evaluation schemes of Section 6.3. This is the subject of the next subsection.

6.5.2 Q-Learning and Approximate Policy Iteration

We will now consider Q-learning methods with linear Q-factor approximation. As we discussed earlier (cf. Fig. 6.5.1), we may view Q-factors as optimal costs of a certain discounted DP problem, whose states are the state-control pairs (i, u). We may thus apply the TD/approximate policy iteration methods of Section 6.3. For this, we need to introduce a linear parametric architecture $\tilde{Q}(i, u, r)$,

$$\tilde{Q}(i, u, r) = \phi(i, u)'r, \tag{6.196}$$

where $\phi(i, u)$ is a feature vector that depends on both state and control.

At the typical iteration, given the current policy μ, these methods find an approximate solution $\tilde{Q}_\mu(i, u, r)$ of the projected equation for Q-factors corresponding to μ, and then obtain a new policy $\overline{\mu}$ by

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \tilde{Q}_\mu(i, u, r).$$

For example, similar to our discussion in Section 6.3.4, LSTD(0) with a linear parametric architecture of the form (6.196) generates a trajectory $\big\{(i_0, u_0), (i_1, u_1), \dots\big\}$ using the current policy $[u_t = \mu(i_t)]$, and finds at time k the unique solution of the projected equation [cf. Eq. (6.53)]

$$\sum_{t=0}^{k} \phi(i_t, u_t) q_{k,t} = 0,$$

where $q_{k,t}$ are the corresponding TD

$$q_{k,t} = \phi(i_t, u_t)'r_k - \alpha\phi(i_{t+1}, u_{t+1})'r_k - g(i_t, u_t, i_{t+1}), \tag{6.197}$$

[cf. Eq. (6.54)]. Also, LSPE(0) is given by [cf. Eq. (6.72)]

$$r_{k+1} = r_k - \frac{\gamma}{k+1} G_k \sum_{t=0}^{k} \phi(i_t, u_t) q_{k,t}, \tag{6.198}$$

where γ is a positive stepsize, G_k is a positive definite matrix, such as

$$G_k = \left(\frac{\beta}{k+1} I + \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t, u_t) \phi(i_t, u_t)' \right)^{-1},$$

with $\beta > 0$, or a diagonal approximation thereof.

There are also optimistic approximate policy iteration methods based on LSPE(0), LSTD(0), and TD(0), similar to the ones we discussed earlier. As an example, let us consider the extreme case of TD(0) that uses a single sample between policy updates. At the start of iteration k, we have the current parameter vector r_k, we are at some state i_k, and we have chosen a control u_k. Then:

(1) We simulate the next transition (i_k, i_{k+1}) using the transition probabilities $p_{i_k j}(u_k)$.

(2) We generate the control u_{k+1} from the minimization

$$u_{k+1} = \arg \min_{u \in U(i_{k+1})} \tilde{Q}(i_{k+1}, u, r_k). \tag{6.199}$$

(3) We update the parameter vector via

$$r_{k+1} = r_k - \gamma_k \phi(i_k, u_k) q_{k,k}, \tag{6.200}$$

where γ_k is a positive stepsize, and $q_{k,k}$ is the TD

$$q_{k,k} = \phi(i_k, u_k)' r_k - \alpha \phi(i_{k+1}, u_{k+1})' r_k - g(i_k, u_k, i_{k+1});$$

[cf. Eq. (6.197)].

The process is now repeated with r_{k+1}, i_{k+1}, and u_{k+1} replacing r_k, i_k, and u_k, respectively.

Exploration

In simulation-based methods, a major concern is the issue of exploration in the approximate policy evaluation step, to ensure that state-control pairs $(i, u) \neq \big(i, \mu(i)\big)$ are generated sufficiently often in the simulation. For this, the exploration-enhanced schemes discussed in Section 6.3.7 may be used in conjunction with LSTD. As an example, given the current policy μ, we may use any exploration-enhanced transition mechanism to generate a sequence $\big\{(i_0, u_0), (i_1, u_1), \ldots\big\}$, and then use LSTD(0) with extra transitions

$$(i_k, u_k) \rightarrow \big(j_k, \mu(j_k)\big),$$

where j_k is generated from (i_k, u_k) using the transition probabilities $p_{i_k j}(u_k)$ (cf. Section 6.3.7). Because LSTD(0) does not require that the underlying mapping ΠT be a contraction, we may design a *single* sequence

$\big\{(i_k, u_k, j_k)\big\}$ that is appropriately exploration-enhanced, and reuse it for all policies generated during policy iteration. This scheme results in substantial economies in simulation overhead. However, it can be used only for $\lambda = 0$, since the simulation samples of multistep policy evaluation methods must depend on the policy.

Alternatively, for $\lambda > 0$, we may use an exploration scheme based on LSTD(λ) with modified temporal differences (cf. Section 6.3.7). In such a scheme, we generate a sequence of state-control pairs $\big\{(i_0, u_0), (i_1, u_1), \ldots\big\}$ according to transition probabilities

$$p_{i_k i_{k+1}}(u_k)\nu(u_{k+1} \mid i_{k+1}),$$

where $\nu(u \mid i)$ is a probability distribution over the control constraint set $U(i)$, which provides a mechanism for exploration. Note that in this case the probability ratios in the modified temporal difference of Eq. (6.124) have the form

$$\frac{p_{i_k i_{k+1}}(u_k)\delta\big(u_{k+1} = \mu(i_{k+1})\big)}{p_{i_k i_{k+1}}(u_k)\nu(u_{k+1} \mid i_{k+1})} = \frac{\delta\big(u_{k+1} = \mu(i_{k+1})\big)}{\nu(u_{k+1} \mid i_{k+1})},$$

and do not depend on the transitions probabilities $p_{i_k i_{k+1}}(u_k)$. Generally, in the context of Q-learning, the required amount of exploration is likely to be substantial, so the underlying mapping $\overline{\Pi}T$ may not be a contraction, in which case the validity of LSPE(λ) or TD(λ) comes into doubt (unless λ is very close to 1), as discussed in Section 6.3.7.

As in other forms of policy iteration, the behavior of all the algorithms described is very complex, involving for example near-singular matrix inversion (cf. Section 6.3.4) or policy oscillations (cf. Section 6.3.8), and there is no guarantee of success (except for general error bounds for approximate policy iteration methods). However, Q-learning with approximate policy iteration is often tried because of its model-free character [it does not require knowledge of $p_{ij}(u)$].

6.5.3 Q-Learning for Optimal Stopping Problems

The policy evaluation algorithms of Section 6.3, such as TD(λ), LSPE(λ), and LSTD(λ), apply when there is a single policy to be evaluated in the context of approximate policy iteration. We may try to extend these methods to the case of multiple policies, by aiming to solve by simulation the projected equation

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

where T is a DP mapping that now involves minimization over multiple controls. However, there are some difficulties:

(a) The mapping ΠT is nonlinear, so a simulation-based approximation approach like LSTD breaks down.

(b) ΠT may not in general be a contraction with respect to any norm, so the PVI iteration

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

[cf. Eq. (6.42)] may diverge and simulation-based LSPE-like approximations may also diverge.

(c) Even if ΠT is a contraction, so the above PVI iteration converges, the simulation-based LSPE-like approximations may not admit an efficient recursive implementation because $T(\Phi r_k)$ is a nonlinear function of r_k.

In this section we discuss the extension of iterative LSPE-type ideas for the special case of an optimal stopping problem where the last two difficulties noted above can be largely overcome. Optimal stopping problems are a special case of DP problems where we can only choose whether to terminate at the current state or not. Examples are problems of search, sequential hypothesis testing, and pricing of derivative financial instruments (see Section 4.4 of Vol. I, and Section 3.4 of the present volume).

We are given a Markov chain with state space $\{1, \ldots, n\}$, described by transition probabilities p_{ij}. We assume that the states form a single recurrent class, so that the steady-state distribution vector $\xi = (\xi_1, \ldots, \xi_n)$ satisfies $\xi_i > 0$ for all i, as in Section 6.3. Given the current state i, we assume that we have two options: to stop and incur a cost $c(i)$, or to continue and incur a cost $g(i, j)$, where j is the next state (there is no control to affect the corresponding transition probabilities). The problem is to minimize the associated α-discounted infinite horizon cost.

We associate a Q-factor with each of the two possible decisions. The Q-factor for the decision to stop is equal to $c(i)$. The Q-factor for the decision to continue is denoted by $Q(i)$, and satisfies Bellman's equation

$$Q(i) = \sum_{j=1}^{n} p_{ij}\Big(g(i,j) + \alpha \min\big\{c(j), Q(j)\big\}\Big). \tag{6.201}$$

The Q-learning algorithm generates an infinitely long sequence of states $\{i_0, i_1, \ldots\}$, with all states generated infinitely often, and a corresponding sequence of transitions $\big\{(i_k, j_k)\big\}$, generated according to the transition probabilities $p_{i_k j}$. It updates the Q-factor for the decision to continue as follows [cf. Eqs. (6.181)-(6.182)]:

$$Q_{k+1}(i) = (1 - \gamma_k)Q_k(i) + \gamma_k(F_k Q_k)(i), \qquad \forall\ i,$$

where the components of the mapping F_k are defined by

$$(F_k Q)(i_k) = g(i_k, j_k) + \alpha \min\big\{c(j_k), Q(j_k)\big\},$$

and

$$(F_k Q)(i) = Q(i), \qquad \forall\ i \neq i_k.$$

The convergence of this algorithm is addressed by the general theory of Q-learning discussed earlier. Once the Q-factors are calculated, an optimal policy can be implemented by stopping at state i if and only if $c(i) \leq Q(i)$. However, when the number of states is very large, the algorithm is impractical, which motivates Q-factor approximations.

Let us introduce the mapping $F : \Re^n \mapsto \Re^n$ given by

$$(FQ)(i) = \sum_{j=1}^{n} p_{ij}\Big(g(i,j) + \alpha \min\big\{c(j), Q(j)\big\}\Big).$$

This mapping can be written in more compact notation as

$$FQ = g + \alpha P f(Q),$$

where g is the vector whose ith component is

$$\sum_{j=1}^{n} p_{ij} g(i,j), \tag{6.202}$$

and $f(Q)$ is the function whose jth component is

$$f_j(Q) = \min\big\{c(j),\, Q(j)\big\}. \tag{6.203}$$

We note that the (exact) Q-factor for the choice to continue is the unique fixed point of F [cf. Eq. (6.201)].

Let $\|\cdot\|_\xi$ be the weighted Euclidean norm associated with the steady-state probability vector ξ. We claim that F is a contraction with respect to this norm. Indeed, for any two vectors Q and $\overline{Q}$, we have

$$\big|(FQ)(i) - (F\overline{Q})(i)\big| \leq \alpha \sum_{j=1}^{n} p_{ij}\big|f_j(Q) - f_j(\overline{Q})\big| \leq \alpha \sum_{j=1}^{n} p_{ij}\big|Q(j) - \overline{Q}(j)\big|,$$

or

$$|FQ - F\overline{Q}| \leq \alpha P|Q - \overline{Q}|,$$

where we use the notation $|x|$ to denote a vector whose components are the absolute values of the components of x. Hence,

$$\|FQ - F\overline{Q}\|_\xi \leq \alpha \big\|P|Q - \overline{Q}|\,\big\|_\xi \leq \alpha \|Q - \overline{Q}\|_\xi,$$

where the last step follows from the inequality $\|PJ\|_\xi \leq \|J\|_\xi$, which holds for every vector J (cf. Lemma 6.3.1). We conclude that F is a contraction with respect to $\|\cdot\|_\xi$, with modulus α.

We will now consider Q-factor approximations, using a linear approximation architecture

$$\tilde{Q}(i,r) = \phi(i)'r,$$

where $\phi(i)$ is an s-dimensional feature vector associated with state i. We also write the vector

$$\big(\tilde{Q}(1,r),\ldots,\tilde{Q}(n,r)\big)'$$

in the compact form Φr, where as in Section 6.3, Φ is the $n \times s$ matrix whose rows are $\phi(i)'$, $i = 1,\ldots,n$. We assume that Φ has rank s, and we denote by Π the projection mapping with respect to $\|\cdot\|_\xi$ on the subspace $S = \{\Phi r \mid r \in \Re^s\}$.

Because F is a contraction with respect to $\|\cdot\|_\xi$ with modulus α, and Π is nonexpansive, the mapping ΠF is a contraction with respect to $\|\cdot\|_\xi$ with modulus α. Therefore, the algorithm

$$\Phi r_{k+1} = \Pi F(\Phi r_k) \tag{6.204}$$

converges to the unique fixed point of ΠF. This is the analog of the PVI algorithm (cf. Section 6.3.2).

As in Section 6.3.2, we can write the PVI iteration (6.204) as

$$r_{k+1} = \arg\min_{r\in\Re^s} \big\|\Phi r - \big(g + \alpha P f(\Phi r_k)\big)\big\|_\xi^2, \tag{6.205}$$

where g and f are defined by Eqs. (6.202) and (6.203). By setting to 0 the gradient of the quadratic function in Eq. (6.205), we see that the iteration is written as

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

where

$$C(r_k) = \Phi'\Xi\big(\Phi r_k - \alpha P f(\Phi r_k)\big), \qquad d = \Phi'\Xi g.$$

Similar to Section 6.3.3, we may implement a simulation-based approximate version of this iteration, thereby obtaining an analog of the LSPE(0) method. In particular, we generate a single infinitely long simulated trajectory $(i_0, i_1, \ldots)$ corresponding to an unstopped system, i.e., using the transition probabilities p_{ij}. Following the transition (i_k, i_{k+1}), we update r_k by

$$r_{k+1} = r_k - \left(\sum_{t=0}^{k} \phi(i_t)\phi(i_t)'\right)^{-1} \sum_{t=0}^{k} \phi(i_t) q_{k,t}, \tag{6.206}$$

where $q_{k,t}$ is the TD,

$$q_{k,t} = \phi(i_t)'r_k - \alpha \min\big\{c(i_{t+1}), \phi(i_{t+1})'r_k\big\} - g(i_t, i_{t+1}). \tag{6.207}$$

Similar to the calculations involving the relation between PVI and LSPE, it can be shown that r_{k+1} as given by this iteration is equal to the iterate produced by the iteration $\Phi r_{k+1} = \Pi F(\Phi r_k)$ plus a simulation-induced error that asymptotically converges to 0 with probability 1 (see the paper

by Yu and Bertsekas [YuB07], to which we refer for further analysis). As a result, the generated sequence $\{\Phi r_k\}$ asymptotically converges to the unique fixed point of ΠF. Note that similar to discounted problems, we may also use a scaled version of the PVI algorithm,

$$r_{k+1} = r_k - \gamma G\big(C(r_k) - d\big), \tag{6.208}$$

where γ is a positive stepsize, and G is a scaling matrix. If G is positive definite symmetric can be shown that this iteration converges to the unique solution of the projected equation if γ is sufficiently small. [The proof of this is somewhat more complicated than the corresponding proof of Section 6.3.2 because $C(r_k)$ depends nonlinearly on r_k. It requires algorithmic analysis using the theory of variational inequalities; see [Ber09b], [Ber11a].] We may approximate the scaled PVI algorithm (6.208) by a simulation-based scaled LSPE version of the form

$$r_{k+1} = r_k - \frac{\gamma}{k+1} G_k \sum_{t=0}^{k} \phi(i_t) q_{k,t},$$

where G_k is a positive definite symmetric matrix and γ is a sufficiently small positive stepsize. For example, we may use a diagonal approximation to the inverse in Eq. (6.206).

In comparing the Q-learning iteration (6.206)-(6.207) with the alternative optimistic LSPE version (6.198), we note that it has considerably higher computation overhead. In the process of updating r_{k+1} via Eq. (6.206), we can compute the matrix $\sum_{t=0}^{k} \phi(i_t)\phi(i_t)'$ and the vector $\sum_{t=0}^{k} \phi(i_t) q_{k,t}$ iteratively as in the LSPE algorithms of Section 6.3. However, the terms

$$\min\big\{c(i_{t+1}), \phi(i_{t+1})' r_k\big\}$$

in the TD formula (6.207) need to be recomputed for all the samples i_{t+1}, $t \leq k$. Intuitively, this computation corresponds to repartitioning the states into those at which to stop and those at which to continue, based on the *current* approximate Q-factors Φr_k. By contrast, in the corresponding optimistic LSPE version (6.198), there is no repartitioning, and these terms are replaced by $\tilde{w}(i_{t+1}, r_k)$, given by

$$\tilde{w}(i_{t+1}, r_k) = \begin{cases} c(i_{t+1}) & \text{if } t \in T, \\ \phi(i_{t+1})' r_k & \text{if } t \notin T, \end{cases}$$

where

$$T = \big\{t \mid c(i_{t+1}) \leq \phi(i_{t+1})' r_t\big\}$$

is the set of states to stop based on the approximate Q-factors Φr_t, calculated at time t (rather than the current time k). In particular, the term

$$\sum_{t=0}^{k} \phi(i_t) \min\big\{c(i_{t+1}), \phi(i_{t+1})' r_k\big\}$$

in Eqs. (6.206), (6.207) is replaced by

$$\sum_{t=0}^{k} \phi(i_t)\tilde{w}(i_{t+1}, r_k) = \sum_{t\le k,\, t\in T} \phi(i_t)c(i_{t+1}) + \left(\sum_{t\le k,\, t\notin T} \phi(i_t)\phi(i_{t+1})'\right) r_k, \tag{6.209}$$

which can be efficiently updated at each time k. It can be seen that the optimistic algorithm that uses the expression (6.209) (no repartitioning) can only converge to the same limit as the nonoptimistic version (6.206). However, there is no convergence proof of this algorithm at present.

Another variant of the algorithm, with a more solid theoretical foundation, is obtained by simply replacing the term $\phi(i_{t+1})' r_k$ in the TD formula (6.207) by $\phi(i_{t+1})' r_t$, thereby eliminating the extra overhead for repartitioning. The idea is that for large k and t, these two terms are close to each other, so convergence is still maintained. The convergence analysis of this algorithm and some variations is based on the theory of stochastic approximation methods, and is given in the paper by Yu and Bertsekas [YuB07] to which we refer for further discussion.

Constrained Policy Iteration and Optimal Stopping

It is natural in approximate DP to try to exploit whatever prior information is available about J^*. In particular, if it is known that J^* belongs to a subset $\mathcal{J}$ of $\Re^n$, we may try to find an approximation Φr that belongs to $\mathcal{J}$. This leads to projected equations involving projection on a restricted subset of the approximation subspace S. Corresponding analogs of the LSTD and LSPE-type methods for such projected equations involve the solution of linear variational inequalities rather linear systems of equations. The details of this are beyond our scope, and we refer to [Ber09b], [Ber11a] for a discussion.

In the practically common case where an upper bound of J^* is available, a simple possibility is to modify the policy iteration algorithm. In particular, suppose that we know a vector $\bar{J}$ with $\bar{J}(i) \ge J^*(i)$ for all i. Then the approximate policy iteration method can be modified to incorporate this knowledge as follows. Given a policy μ, we evaluate it by finding an approximation $\Phi\tilde{r}_\mu$ to the solution $\tilde{J}_\mu$ of the equation

$$\tilde{J}_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g(i,\mu(i),j) + \alpha \min\big\{\bar{J}(j),\, \tilde{J}_\mu(j)\big\}\Big), \qquad i = 1, \ldots, n, \tag{6.210}$$

followed by the (modified) policy improvement

$$\bar{\mu}(i) = \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha \min\big\{\bar{J}(j), \phi(j)'\tilde{r}_\mu\big\}\Big), \qquad i = 1,\ldots,n, \tag{6.211}$$

where $\phi(j)'$ is the row of Φ that corresponds to state j.

Note that Eq. (6.210) is Bellman's equation for the Q-factor of an optimal stopping problem that involves the stopping cost $\bar{J}(i)$ at state i [cf. Eq. (6.201)]. Under the assumption $\bar{J}(i) \geq J^*(i)$ for all i, and a lookup table representation ($\Phi = I$), it can be shown that the method (6.210)-(6.211) yields J^* in a finite number of iterations, just like the standard (exact) policy iteration method (Exercise 6.17). When a compact feature-based representation is used ($\Phi \neq I$), the approximate policy evaluation based on Eq. (6.210) can be performed using the Q-learning algorithms described earlier in this section. The method may exhibit oscillatory behavior and is subject to chattering, similar to its unconstrained policy iteration counterpart (cf. Section 6.3.8).

6.5.4 Finite-Horizon Q-Learning

We will now briefly discuss Q-learning and related approximations for finite-horizon problems. We will emphasize on-line algorithms that are suitable for relatively short horizon problems. Such problems are additionally important because they arise in the context of multistep lookahead and rolling horizon schemes, possibly with cost function approximation at the end of the horizon.

One may develop extensions of the Q-learning algorithms of the preceding sections to deal with finite horizon problems, with or without cost function approximation. For example, one may easily develop versions of the projected Bellman equation, and corresponding LSTD and LSPE-type algorithms (see the end-of-chapter exercises). However, with a finite horizon, there are a few alternative approaches, with an on-line character, which resemble rollout algorithms. In particular, at state-time pair (i_k, k), we may compute approximate Q-factors

$$\tilde{Q}_k(i_k, u_k), \qquad u_k \in U_k(i_k),$$

and use on-line the control $\tilde{u}_k \in U_k(i_k)$ that minimizes $\tilde{Q}_k(i_k, u_k)$ over $u_k \in U_k(i_k)$. The approximate Q-factors have the form

$$\tilde{Q}_k(i_k, u_k) = \sum_{i_{k+1}=1}^{n_k} p_{i_k i_{k+1}}(u_k)\Big\{g(i_k, u_k, i_{k+1}) + \min_{u_{k+1}\in U_{k+1}(i_{k+1})} \hat{Q}_{k+1}(i_{k+1}, u_{k+1})\Big\}, \tag{6.212}$$

where $\hat{Q}_{k+1}$ may be computed in a number of ways:

(1) $\hat{Q}_{k+1}$ may be the cost function $\tilde{J}_{k+1}$ of a base heuristic (and is thus independent of u_{k+1}), in which case Eq. (6.212) takes the form

$$\tilde{Q}_k(i_k, u_k) = \sum_{i_{k+1}=1}^{n_k} p_{i_k i_{k+1}}(u_k)\big(g(i_k, u_k, i_{k+1}) + \tilde{J}_{k+1}(i_{k+1})\big). \tag{6.213}$$

This is the rollout algorithm discussed at length in Chapter 6 of Vol. I. A variation is when multiple base heuristics are used and $\tilde{J}_{k+1}$ is the minimum of the cost functions of these heuristics. These schemes may also be combined with a rolling and/or limited lookahead horizon.

(2) $\hat{Q}_{k+1}$ is an approximately optimal cost function $\tilde{J}_{k+1}$ [independent of u_{k+1} as in Eq. (6.213)], which is computed by (possibly multistep lookahead or rolling horizon) DP based on limited sampling to approximate the various expected values arising in the DP algorithm. Thus, here the function $\tilde{J}_{k+1}$ of Eq. (6.213) corresponds to a (finite-horizon) near-optimal policy in place of the base policy used by rollout. These schemes are well suited for problems with a large (or infinite) state space but only a small number of controls per state, and may also involve selective pruning of the control constraint set to reduce the associated DP computations. The book by Chang, Fu, Hu, and Marcus [CFH07] has extensive discussions of approaches of this type, including systematic forms of *adaptive sampling* that aim to reduce the effects of limited simulation (less sampling for controls that seem less promising at a given state, and less sampling for future states that are less likely to be visited starting from the current state i_k).

(3) $\hat{Q}_{k+1}$ is computed using a linear parametric architecture of the form

$$\hat{Q}_{k+1}(i_{k+1}, u_{k+1}) = \phi(i_{k+1}, u_{k+1})' r_{k+1}, \tag{6.214}$$

where r_{k+1} is a parameter vector. In particular, $\hat{Q}_{k+1}$ may be obtained by a least-squares fit/regression or interpolation based on values computed at a subset of selected state-control pairs (cf. Section 6.4.3 of Vol. I). These values may be computed by finite horizon rollout, using as base policy the greedy policy corresponding to the preceding approximate Q-values in a backwards (off-line) Q-learning scheme:

$$\tilde{\mu}_i(x_i) = \arg\min_{u_i \in U_i(x_i)} \hat{Q}_i(x_i, u_i), \qquad i = k+2, \ldots, N-1. \tag{6.215}$$

Thus, in such a scheme, we first compute

$$\tilde{Q}_{N-1}(i_{N-1}, u_{N-1}) = \sum_{i_N=1}^{n_N} p_{i_{N-1} i_N}(u_{N-1})\big\{g(i_{N-1}, u_{N-1}, i_N) + J_N(i_N)\big\}$$

by the final stage DP computation at a subset of selected state-control pairs (i_{N-1}, u_{N-1}), followed by a least squares fit of the obtained values to obtain $\hat{Q}_{N-1}$ in the form (6.214); then we compute $\tilde{Q}_{N-2}$ at a subset of selected state-control pairs (i_{N-2}, u_{N-2}) by rollout using the base policy $\{\tilde{\mu}_{N-1}\}$ defined by Eq. (6.215), followed by a least squares fit of the obtained values to obtain $\hat{Q}_{N-2}$ in the form (6.214); then compute $\tilde{Q}_{N-3}$ at a subset of selected state-control pairs (i_{N-3}, u_{N-3}) by rollout using the base policy $\{\tilde{\mu}_{N-2}, \tilde{\mu}_{N-1}\}$ defined by Eq. (6.215), etc.

One advantage of finite horizon formulations is that convergence issues of the type arising in policy or value iteration methods do not play a significant role, so anomalous behavior does not arise. This is, however, a mixed blessing as it may mask poor performance and/or important qualitative differences between alternative approaches.

6.6 STOCHASTIC SHORTEST PATH PROBLEMS

In this section we consider policy evaluation for finite-state stochastic shortest path (SSP) problems (cf. Chapter 2). We assume that there is no discounting ($\alpha = 1$), and that the states are $0, 1, \ldots, n$, where state 0 is a special cost-free termination state. We focus on a fixed proper policy μ, under which all the states $1, \ldots, n$ are transient.

There are natural extensions of the LSTD(λ) and LSPE(λ) algorithms. We introduce a linear approximation architecture of the form

$$\tilde{J}(i, r) = \phi(i)'r, \qquad i = 0, 1, \ldots, n,$$

and the subspace

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

where, as in Section 6.3, Φ is the $n \times s$ matrix whose rows are $\phi(i)'$, $i = 1, \ldots, n$. We assume that Φ has rank s. Also, for notational convenience in the subsequent formulas, we define $\phi(0) = 0$.

The algorithms use a sequence of simulated trajectories, each of the form $(i_0, i_1, \ldots, i_N)$, where $i_N = 0$, and $i_t \neq 0$ for $t < N$. Once a trajectory is completed, an initial state i_0 for the next trajectory is chosen according to a fixed probability distribution $q_0 = \big(q_0(1), \ldots, q_0(n)\big)$, where

$$q_0(i) = P(i_0 = i), \qquad i = 1, \ldots, n, \tag{6.216}$$

and the process is repeated.

For a trajectory $i_0, i_1, \ldots$, of the SSP problem consider the probabilities

$$q_t(i) = P(i_t = i), \qquad i = 1, \ldots, n, \ t = 0, 1, \ldots$$

Note that $q_t(i)$ diminishes to 0 as $t \to \infty$ at the rate of a geometric progression (cf. Section 2.1), so the limits

$$q(i) = \sum_{t=0}^{\infty} q_t(i), \qquad i = 1, \ldots, n,$$

are finite. Let q be the vector with components $q(1), \ldots, q(n)$. We assume that $q_0(i)$ are chosen so that $q(i) > 0$ for all i [a stronger assumption is that $q_0(i) > 0$ for all i]. We introduce the norm

$$\|J\|_q = \sqrt{\sum_{i=1}^{n} q(i)\big(J(i)\big)^2},$$

and we denote by Π the projection onto the subspace S with respect to this norm. In the context of the SSP problem, the projection norm $\|\cdot\|_q$ plays a role similar to the one played by the steady-state distribution norm $\|\cdot\|_\xi$ for discounted problems (cf. Section 6.3).

Let P be the $n \times n$ matrix with components p_{ij}, $i, j = 1, \ldots, n$. Consider also the mapping $T : \Re^n \mapsto \Re^n$ given by

$$TJ = g + PJ,$$

where g is the vector with components $\sum_{j=0}^{n} p_{ij} g(i,j)$, $i = 1, \ldots, n$. For $\lambda \in [0, 1)$, define the mapping

$$T^{(\lambda)} = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t T^{t+1}$$

[cf. Eq. (6.84)]. Similar to Section 6.3, we have

$$T^{(\lambda)} J = P^{(\lambda)} J + (I - \lambda P)^{-1} g,$$

where

$$P^{(\lambda)} = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t P^{t+1} \tag{6.217}$$

[cf. Eq. (6.85)].

We will now show that $\Pi T^{(\lambda)}$ is a contraction, so that it has a unique fixed point.

Proposition 6.6.1: For all $\lambda \in [0, 1)$, $\Pi T^{(\lambda)}$ is a contraction with respect to some norm.

Proof: Let $\lambda > 0$. We will show that $T^{(\lambda)}$ is a contraction with respect to the projection norm $\|\cdot\|_q$, so the same is true for $\Pi T^{(\lambda)}$, since Π is nonexpansive. Let us first note that with an argument like the one in the proof of Lemma 6.3.1, we can show that

$$\|PJ\|_q \leq \|J\|_q, \qquad J \in \Re^n.$$

Indeed, we have $q = \sum_{t=0}^{\infty} q_t$ and $q'_{t+1} = q'_t P$, so

$$q'P = \sum_{t=0}^{\infty} q'_t P = \sum_{t=1}^{\infty} q'_t = q' - q'_0,$$

or

$$\sum_{i=1}^{n} q(i)p_{ij} = q(j) - q_0(j), \qquad \forall\ j.$$

Using this relation, we have for all $J \in \Re^n$,

$$\begin{aligned} \|PJ\|_q^2 &= \sum_{i=1}^{n} q(i) \left(\sum_{j=1}^{n} p_{ij} J(j) \right)^2 \\ &\leq \sum_{i=1}^{n} q(i) \sum_{j=1}^{n} p_{ij} J(j)^2 \\ &= \sum_{j=1}^{n} J(j)^2 \sum_{i=1}^{n} q(i)p_{ij} \\ &= \sum_{j=1}^{n} \big(q(j) - q_0(j)\big) J(j)^2 \\ &\leq \|J\|_q^2. \end{aligned} \tag{6.218}$$

From the relation $\|PJ\|_q \leq \|J\|_q$ it follows that

$$\|P^t J\|_q \leq \|J\|_q, \qquad J \in \Re^n,\ t = 0, 1, \ldots$$

Thus, by using the definition (6.217) of $P^{(\lambda)}$, we also have

$$\|P^{(\lambda)} J\|_q \leq \|J\|_q, \qquad J \in \Re^n.$$

Since $\lim_{t\to\infty} P^t J = 0$ for any $J \in \Re^n$, it follows that $\|P^t J\|_q < \|J\|_q$ for all $J \neq 0$ and t sufficiently large. Therefore,

$$\|P^{(\lambda)} J\|_q < \|J\|_q, \qquad \text{for all } J \neq 0. \tag{6.219}$$

We now define

$$\beta = \max\big\{ \|P^{(\lambda)} J\|_q \mid \|J\|_q = 1 \big\}$$

and note that since the maximum in the definition of β is attained by the Weierstrass Theorem (a continuous function attains a maximum over a compact set), we have $\beta < 1$ in view of Eq. (6.219). Since

$$\|P^{(\lambda)}J\|_q \leq \beta \|J\|_q, \qquad J \in \Re^n,$$

it follows that $P^{(\lambda)}$ is a contraction of modulus β with respect to $\|\cdot\|_q$.

Let $\lambda = 0$. We use a different argument because T is not necessarily a contraction with respect to $\|\cdot\|_q$. [An example is given following Prop. 6.8.2. Note also that if $q_0(i) > 0$ for all i, from the calculation of Eq. (6.218) it follows that P and hence T is a contraction with respect to $\|\cdot\|_q$.] We show that ΠT is a contraction with respect to a different norm by showing that the eigenvalues of ΠP lie strictly within the unit circle.† Indeed, with an argument like the one used to prove Lemma 6.3.1, we have $\|PJ\|_q \leq \|J\|_q$ for all J, which implies that $\|\Pi PJ\|_q \leq \|J\|_q$, so the eigenvalues of ΠP cannot be outside the unit circle. Assume to arrive at a contradiction that ν is an eigenvalue of ΠP with $|\nu| = 1$, and let ζ be a corresponding eigenvector. We claim that $P\zeta$ must have both real and imaginary components in the subspace S. If this were not so, we would have $P\zeta \neq \Pi P\zeta$, so that

$$\|P\zeta\|_q > \|\Pi P\zeta\|_q = \|\nu\zeta\|_q = |\nu| \, \|\zeta\|_q = \|\zeta\|_q,$$

which contradicts the fact $\|PJ\|_q \leq \|J\|_q$ for all J. Thus, the real and imaginary components of $P\zeta$ are in S, which implies that $P\zeta = \Pi P\zeta = \nu\zeta$, so that ν is an eigenvalue of P. This is a contradiction because $|\nu| = 1$ while the eigenvalues of P are strictly within the unit circle, since the policy being evaluated is proper. **Q.E.D.**

The preceding proof has shown that $\Pi T^{(\lambda)}$ is a contraction with respect to $\|\cdot\|_q$ when $\lambda > 0$. As a result, similar to Prop. 6.3.5, we can obtain the error bound

$$\|J_\mu - \Phi r^*_\lambda\|_q \leq \frac{1}{\sqrt{1-\alpha^2_\lambda}} \, \|J_\mu - \Pi J_\mu\|_q, \qquad \lambda > 0,$$

† We use here the fact that if a square matrix has eigenvalues strictly within the unit circle, then there exists a norm with respect to which the linear mapping defined by the matrix is a contraction. Also in the following argument, the projection Πz of a complex vector z is obtained by separately projecting the real and the imaginary components of z on S. The projection norm for a complex vector $x + iy$ is defined by

$$\|x + iy\|_q = \sqrt{\|x\|^2_q + \|y\|^2_q}.$$

where Φr_λ^* and α_λ are the fixed point and contraction modulus of $\Pi T^{(\lambda)}$, respectively. When $\lambda = 0$, we have

$$\begin{aligned}\|J_\mu - \Phi r_0^*\| &\leq \|J_\mu - \Pi J_\mu\| + \|\Pi J_\mu - \Phi r_0^*\| \\ &= \|J_\mu - \Pi J_\mu\| + \|\Pi T J_\mu - \Pi T(\Phi r_0^*)\| \\ &= \|J_\mu - \Pi J_\mu\| + \alpha_0 \|J_\mu - \Phi r_0^*\|,\end{aligned}$$

where $\|\cdot\|$ is the norm with respect to which ΠT is a contraction (cf. Prop. 6.6.1), and Φr_0^* and α_0 are the fixed point and contraction modulus of ΠT. We thus have the error bound

$$\|J_\mu - \Phi r_0^*\| \leq \frac{1}{1-\alpha_0} \|J_\mu - \Pi J_\mu\|.$$

Similar to the discounted problem case, the projected equation can be written as a linear equation of the form $Cr = d$. The corresponding LSTD and LSPE algorithms use simulation-based approximations C_k and d_k. This simulation generates a sequence of trajectories of the form $(i_0, i_1, \ldots, i_N)$, where $i_N = 0$, and $i_t \neq 0$ for $t < N$. Once a trajectory is completed, an initial state i_0 for the next trajectory is chosen according to a fixed probability distribution $q_0 = \big(q_0(1), \ldots, q_0(n)\big)$. The LSTD method approximates the solution $C^{-1}d$ of the projected equation by $C_k^{-1} d_k$, where C_k and d_k are simulation-based approximations to C and d, respectively. The LSPE algorithm and its scaled versions are defined by

$$r_{k+1} = r_k - \gamma G_k (C_k r_k - d_k),$$

where γ is a sufficiently small stepsize and G_k is a scaling matrix. The derivation of the detailed equations is straightforward but somewhat tedious, and will not be given (see also the discussion in Section 6.8).

Regarding exploration, let us note that the ideas of Sections 6.3.6 and 6.3.7 apply to policy iteration methods for SSP problems. However, because the distribution q_0 for the initial state of the simulated trajectories can be chosen arbitrarily, the problem of exploration may be far less acute in SSP problems than in discounted problems, particularly when simulated trajectories tend to be short. In this case, one may explore various parts of the state space naturally through the restart mechanism, similar to the exploration-enhanced LSPE(λ) and LSTD(λ) methods.

6.7 AVERAGE COST PROBLEMS

In this section we consider average cost problems and related approximations: policy evaluation algorithms such as LSTD(λ) and LSPE(λ), approximate policy iteration, and Q-learning. We assume throughout the finite state model of Section 4.1, with the optimal average cost being the same for all initial states (cf. Section 4.2).

6.7.1 Approximate Policy Evaluation

Let us consider the problem of approximate evaluation of a stationary policy μ. As in the discounted case (Section 6.3), we consider a stationary finite-state Markov chain with states $i = 1, \ldots, n$, transition probabilities p_{ij}, $i, j = 1, \ldots, n$, and stage costs $g(i, j)$. We assume that the states form a single recurrent class. An equivalent way to express this assumption is the following.

Assumption 6.7.1: The Markov chain has a steady-state probability vector $\xi = (\xi_1, \ldots, \xi_n)$ with positive components, i.e., for all $i = 1, \ldots, n$,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} P(i_k = j \mid i_0 = i) = \xi_j > 0, \qquad j = 1, \ldots, n.$$

From Section 4.2, we know that under Assumption 6.7.1, the average cost, denoted by η, is independent of the initial state

$$\eta = \lim_{N\to\infty} \frac{1}{N} E\left\{ \sum_{k=0}^{N-1} g\big(x_k, x_{k+1}\big) \;\Big|\; x_0 = i \right\}, \qquad i = 1, \ldots, n, \tag{6.220}$$

and satisfies

$$\eta = \xi' g,$$

where g is the vector whose ith component is the expected stage cost $\sum_{j=1}^{n} p_{ij} g(i, j)$. (In Chapter 4 we denoted the average cost by λ, but in the present chapter, with apologies to the readers, we reserve λ for use in the TD, LSPE, and LSTD algorithms, hence the change in notation.) Together with a differential cost vector $h = \big(h(1), \ldots, h(n)\big)'$, the average cost η satisfies Bellman's equation

$$h(i) = \sum_{j=1}^{n} p_{ij} \big(g(i, j) - \eta + h(j)\big), \qquad i = 1, \ldots, n.$$

The solution is unique up to a constant shift for the components of h, and can be made unique by eliminating one degree of freedom, such as fixing the differential cost of a single state to 0 (cf. Prop. 4.2.4).

We consider a linear architecture for the differential costs of the form

$$\tilde{h}(i, r) = \phi(i)' r, \qquad i = 1, \ldots, n.$$

where $r \in \Re^s$ is a parameter vector and $\phi(i)$ is a feature vector associated with state i. These feature vectors define the subspace

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

where as in Section 6.3, Φ is the $n \times s$ matrix whose rows are $\phi(i)'$, $i = 1, \ldots, n$. We will thus aim to approximate h by a vector in S, similar to Section 6.3, which dealt with cost approximation in the discounted case.

We introduce the mapping $F : \Re^n \mapsto \Re^n$ defined by

$$FJ = g - \eta e + PJ,$$

where P is the transition probability matrix and e is the unit vector. Note that the definition of F uses the *exact* average cost η, as given by Eq. (6.220). With this notation, Bellman's equation becomes

$$h = Fh,$$

so if we know η, we can try to find or approximate a fixed point of F.

Similar to Section 6.3, we introduce the projected equation

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

where Π is projection on the subspace S with respect to the norm $\|\cdot\|_\xi$. An important issue is whether ΠF is a contraction. For this it is necessary to make the following assumption.

Assumption 6.7.2: The columns of the matrix Φ together with the unit vector $e = (1, \ldots, 1)'$ form a linearly independent set of vectors.

Note the difference with the corresponding Assumption 6.3.2 for the discounted case in Section 6.3. Here, in addition to Φ having rank s, we require that e does not belong to the subspace S. To get a sense why this is needed, observe that if $e \in S$, then ΠF cannot be a contraction, since any scalar multiple of e when added to a fixed point of ΠF would also be a fixed point.

We also consider multistep versions of the projected equation of the form

$$\Phi r = \Pi F^{(\lambda)}(\Phi r), \tag{6.221}$$

where

$$F^{(\lambda)} = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t F^{t+1}.$$

In matrix notation, the mapping $F^{(\lambda)}$ can be written as

$$F^{(\lambda)}J = (1-\lambda)\sum_{t=0}^{\infty}\lambda^t P^{t+1}J + \sum_{t=0}^{\infty}\lambda^t P^t(g-\eta e),$$

or more compactly as

$$F^{(\lambda)}J = P^{(\lambda)}J + (I-\lambda P)^{-1}(g-\eta e), \tag{6.222}$$

where the matrix $P^{(\lambda)}$ is defined by

$$P^{(\lambda)} = (1-\lambda)\sum_{t=0}^{\infty}\lambda^t P^{t+1} \tag{6.223}$$

[cf. Eq. (6.85)]. Note that for $\lambda = 0$, we have $F^{(0)} = F$ and $P^{(0)} = P$.

We wish to delineate conditions under which the mapping $\Pi F^{(\lambda)}$ is a contraction. The following proposition relates to the composition of general linear mappings with Euclidean projections, and captures the essence of our analysis.

Proposition 6.7.1: Let S be a subspace of $\Re^n$ and let $L : \Re^n \mapsto \Re^n$ be a linear mapping,

$$L(x) = Ax + b,$$

where A is an $n \times n$ matrix and b is a vector in $\Re^n$. Let $\|\cdot\|$ be a weighted Euclidean norm with respect to which L is nonexpansive, and let Π denote projection onto S with respect to that norm.

(a) ΠL has a unique fixed point if and only if either 1 is not an eigenvalue of A, or else the eigenvectors corresponding to the eigenvalue 1 do not belong to S.

(b) If ΠL has a unique fixed point, then for all $\gamma \in (0,1)$, the mapping

$$H_\gamma = (1-\gamma)I + \gamma\Pi L$$

is a contraction, i.e., for some scalar $\rho_\gamma \in (0,1)$, we have

$$\|H_\gamma x - H_\gamma y\| \leq \rho_\gamma \|x-y\|, \qquad \forall\ x, y \in \Re^n.$$

Proof: (a) Assume that ΠL has a unique fixed point, or equivalently (in view of the linearity of L) that 0 is the unique fixed point of ΠA. If 1 is an eigenvalue of A with a corresponding eigenvector z that belongs to S, then $Az = z$ and $\Pi Az = \Pi z = z$. Thus, z is a fixed point of ΠA with

$z \neq 0$, a contradiction. Hence, either 1 is not an eigenvalue of A, or else the eigenvectors corresponding to the eigenvalue 1 do not belong to S.

Conversely, assume that either 1 is not an eigenvalue of A, or else the eigenvectors corresponding to the eigenvalue 1 do not belong to S. We will show that the mapping $\Pi(I - A)$ is one-to-one from S to S, and hence the fixed point of ΠL is the unique vector $x^* \in S$ satisfying $\Pi(I - A)x^* = \Pi b$. Indeed, assume the contrary, i.e., that $\Pi(I - A)$ has a nontrivial nullspace in S, so that some $z \in S$ with $z \neq 0$ is a fixed point of ΠA. Then, either $Az = z$ (which is impossible since then 1 is an eigenvalue of A, and z is a corresponding eigenvector that belongs to S), or $Az \neq z$, in which case Az differs from its projection ΠAz and

$$\|z\| = \|\Pi Az\| < \|Az\| \leq \|A\| \, \|z\|,$$

so that $1 < \|A\|$ (which is impossible since L is nonexpansive, and therefore $\|A\| \leq 1$), thereby arriving at a contradiction.

(b) If $z \in \Re^n$ with $z \neq 0$ and $z \neq a\Pi Az$ for all $a \geq 0$, we have

$$\|(1-\gamma)z + \gamma\Pi Az\| < (1-\gamma)\|z\| + \gamma\|\Pi Az\| \leq (1-\gamma)\|z\| + \gamma\|z\| = \|z\|, \tag{6.224}$$

where the strict inequality follows from the strict convexity of the norm, and the weak inequality follows from the non-expansiveness of ΠA. If on the other hand $z \neq 0$ and $z = a\Pi Az$ for some $a \geq 0$, we have $\|(1-\gamma)z + \gamma\Pi Az\| < \|z\|$ because then ΠL has a unique fixed point so $a \neq 1$, and ΠA is nonexpansive so $a < 1$. If we define

$$\rho_\gamma = \sup\{\|(1-\gamma)z + \gamma\Pi Az\| \mid \|z\| \leq 1\},$$

and note that the supremum above is attained by the Weierstrass Theorem (a continuous function attains a minimum over a compact set), we see that Eq. (6.224) yields $\rho_\gamma < 1$ and

$$\|(1-\gamma)z + \gamma\Pi Az\| \leq \rho_\gamma\|z\|, \qquad z \in \Re^n.$$

By letting $z = x - y$, with $x, y \in \Re^n$, and by using the definition of H_γ, we have

$$H_\gamma x - H_\gamma y = H_\gamma(x-y) = (1-\gamma)(x-y) + \gamma\Pi A(x-y) = (1-\gamma)z + \gamma\Pi Az,$$

so by combining the preceding two relations, we obtain

$$\|H_\gamma x - H_\gamma y\| \leq \rho_\gamma\|x - y\|, \qquad x, y \in \Re^n.$$

Q.E.D.

We can now derive the conditions under which the mapping underlying the LSPE iteration is a contraction with respect to $\|\cdot\|_\xi$.

Proposition 6.7.2: The mapping

$$F_{\gamma,\lambda} = (1-\gamma)I + \gamma\Pi F^{(\lambda)}$$

is a contraction with respect to $\|\cdot\|_\xi$ if one of the following is true:

(i) $\lambda \in (0,1)$ and $\gamma \in (0,1]$,

(ii) $\lambda = 0$ and $\gamma \in (0,1)$.

Proof: Consider first the case, $\gamma = 1$ and $\lambda \in (0,1)$. Then $F^{(\lambda)}$ is a linear mapping involving the matrix $P^{(\lambda)}$. Since $0 < \lambda$ and all states form a single recurrent class, all entries of $P^{(\lambda)}$ are positive. Thus $P^{(\lambda)}$ can be expressed as a convex combination

$$P^{(\lambda)} = (1-\beta)I + \beta\bar{P}$$

for some $\beta \in (0,1)$, where $\bar{P}$ is a stochastic matrix with positive entries. We make the following observations:

(i) $\bar{P}$ corresponds to a nonexpansive mapping with respect to the norm $\|\cdot\|_\xi$. The reason is that the steady-state distribution of $\bar{P}$ is ξ [as can be seen by multiplying the relation $P^{(\lambda)} = (1-\beta)I + \beta\bar{P}$ with ξ, and by using the relation $\xi' = \xi' P^{(\lambda)}$ to verify that $\xi' = \xi'\bar{P}$]. Thus, we have $\|\bar{P}z\|_\xi \leq \|z\|_\xi$ for all $z \in \Re^n$ (cf. Lemma 6.3.1), implying that $\bar{P}$ has the nonexpansiveness property mentioned.

(ii) Since $\bar{P}$ has positive entries, the states of the Markov chain corresponding to $\bar{P}$ form a single recurrent class. If z is an eigenvector of $\bar{P}$ corresponding to the eigenvalue 1, we have $z = \bar{P}^k z$ for all $k \geq 0$, so $z = \bar{P}^* z$, where

$$\bar{P}^* = \lim_{N\to\infty} (1/N) \sum_{k=0}^{N-1} \bar{P}^k$$

(cf. Prop. 4.1.2). The rows of $\bar{P}^*$ are all equal to ξ' since the steady-state distribution of $\bar{P}$ is ξ, so the equation $z = \bar{P}^* z$ implies that z is a nonzero multiple of e. Using Assumption 6.7.2, it follows that z does not belong to the subspace S, and from Prop. 6.7.1 (with $\bar{P}$ in place of C, and β in place of γ), we see that $\Pi P^{(\lambda)}$ is a contraction with respect to the norm $\|\cdot\|_\xi$. This implies that $\Pi F^{(\lambda)}$ is also a contraction.

Consider next the case, $\gamma \in (0,1)$ and $\lambda \in (0,1)$. Since $\Pi F^{(\lambda)}$ is a contraction with respect to $\|\cdot\|_\xi$, as just shown, we have for any $J, \bar{J} \in \Re^n$,

$$\begin{aligned}\|F_{\gamma,\lambda}J - F_{\gamma,\lambda}\bar{J}\|_\xi &\leq (1-\gamma)\|J-\bar{J}\|_\xi + \gamma\left\|\Pi F^{(\lambda)}J - \Pi F^{(\lambda)}\bar{J}\right\|_\xi \\ &\leq (1-\gamma+\gamma\beta)\|J-\bar{J}\|_\xi,\end{aligned}$$

where β is the contraction modulus of $F^{(\lambda)}$. Hence, $F_{\gamma,\lambda}$ is a contraction.

Finally, consider the case $\gamma \in (0,1)$ and $\lambda = 0$. We will show that the mapping ΠF has a unique fixed point, by showing that either 1 is not an eigenvalue of P, or else the eigenvectors corresponding to the eigenvalue 1 do not belong to S [cf. Prop. 6.7.1(a)]. Assume the contrary, i.e., that some $z \in S$ with $z \neq 0$ is an eigenvector corresponding to 1. We then have $z = Pz$. From this it follows that $z = P^k z$ for all $k \geq 0$, so $z = P^* z$, where

$$P^* = \lim_{N\to\infty} (1/N) \sum_{k=0}^{N-1} P^k$$

(cf. Prop. 4.1.2). The rows of P^* are all equal to ξ', so the equation $z = P^* z$ implies that z is a nonzero multiple of e. Hence, by Assumption 6.7.2, z cannot belong to S - a contradiction. Thus ΠF has a unique fixed point, and the contraction property of $F_{\gamma,\lambda}$ for $\gamma \in (0,1)$ and $\lambda = 0$ follows from Prop. 6.7.1(b). **Q.E.D.**

Error Estimate

We have shown that for each $\lambda \in [0,1)$, there is a vector Φr^*_λ, the unique fixed point of $\Pi F_{\gamma,\lambda}$, $\gamma \in (0,1)$, which is the limit of LSPE(λ) (cf. Prop. 6.7.2). Let h be any differential cost vector, and let $\beta_{\gamma,\lambda}$ be the modulus of contraction of $\Pi F_{\gamma,\lambda}$, with respect to $\|\cdot\|_\xi$. Similar to the proof of Prop. 6.3.2 for the discounted case, we have

$$\begin{aligned}
\|h - \Phi r^*_\lambda\|^2_\xi &= \|h - \Pi h\|^2_\xi + \|\Pi h - \Phi r^*_\lambda\|^2_\xi \\
&= \|h - \Pi h\|^2_\xi + \left\|\Pi F_{\gamma,\lambda} h - \Pi F_{\gamma,\lambda}(\Phi r^*_\lambda)\right\|^2_\xi \\
&\leq \|h - \Pi h\|^2_\xi + \beta_{\gamma,\lambda} \|h - \Phi r^*_\lambda\|^2_\xi.
\end{aligned}$$

It follows that

$$\|h - \Phi r^*_\lambda\|_\xi \leq \frac{1}{\sqrt{1-\beta^2_{\gamma,\lambda}}} \|h - \Pi h\|_\xi, \qquad \lambda \in [0,1),\ \gamma \in (0,1), \quad (6.225)$$

for all differential cost vector vectors h.

This estimate is a little peculiar because the differential cost vector is not unique. The set of differential cost vectors is

$$D = \{h^* + \gamma e \mid \gamma \in \Re\},$$

where h^* is the bias of the policy evaluated (cf. Section 4.1, and Props. 4.1.1 and 4.1.2). In particular, h^* is the unique $h \in D$ that satisfies $\xi' h = 0$ or equivalently $P^* h = 0$, where

$$P^* = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} P^k.$$

Usually, in average cost policy evaluation, we are interested in obtaining a small error $(h - \Phi r_\lambda^*)$ with the choice of h being immaterial (see the discussion of the next section on approximate policy iteration). It follows that since the estimate (6.225) holds for all $h \in D$, a better error bound can be obtained by using an optimal choice of h in the left-hand side and an optimal choice of γ in the right-hand side. Indeed, Tsitsiklis and Van Roy [TsV99a] have obtained such an optimized error estimate. It has the form

$$\min_{h \in D} \|h - \Phi r_\lambda^*\|_\xi = \left\|h^* - (I - P^*)\Phi r_\lambda^*\right\|_\xi \leq \frac{1}{\sqrt{1-\alpha_\lambda^2}} \|\Pi^* h^* - h^*\|_\xi, \quad (6.226)$$

where h^* is the bias vector, Π^* denotes projection with respect to $\|\cdot\|_\xi$ onto the subspace

$$S^* = \big\{(I - P^*)y \mid y \in S\big\},$$

and α_λ is the minimum over $\gamma \in (0,1)$ of the contraction modulus of the mapping $\Pi^* F_{\gamma,\lambda}$:

$$\alpha_\lambda = \min_{\gamma \in (0,1)} \max_{\|y\|_\xi = 1} \|\Pi^* P_{\gamma,\lambda}\, y\|_\xi,$$

where $P_{\gamma,\lambda} = (1-\gamma)I + \gamma \Pi^* P^{(\lambda)}$. Note that this error bound has similar form with the one for discounted problems (cf. Prop. 6.3.5), but S has been replaced by S^* and Π has been replaced by Π^*. It can be shown that the scalar α_λ decreases as λ increases, and approaches 0 as $\lambda \uparrow 1$. This is consistent with the corresponding error bound for discounted problems (cf. Prop. 6.3.5), and is also consistent with empirical observations, which suggest that smaller values of λ lead to larger approximation errors.

Figure 6.7.1 illustrates and explains the projection operation Π^*, the distance of the bias h^* from its projection $\Pi^* h^*$, and the other terms in the error bound (6.226).

LSTD(λ) and LSPE(λ)

The LSTD(λ) and LSPE(λ) algorithms for average cost are straightforward extensions of the discounted versions, and will only be summarized. The LSTD(λ) algorithm is given by

$$\hat{r}_k = C_k^{-1} d_k.$$

There is also a regression-based version that is well-suited for cases where C_k is nearly singular (cf. Section 6.3.4). The LSPE(λ) iteration can be written (similar to the discounted case) as

$$r_{k+1} = r_k - \gamma G_k (C_k r_k - d_k), \quad (6.227)$$

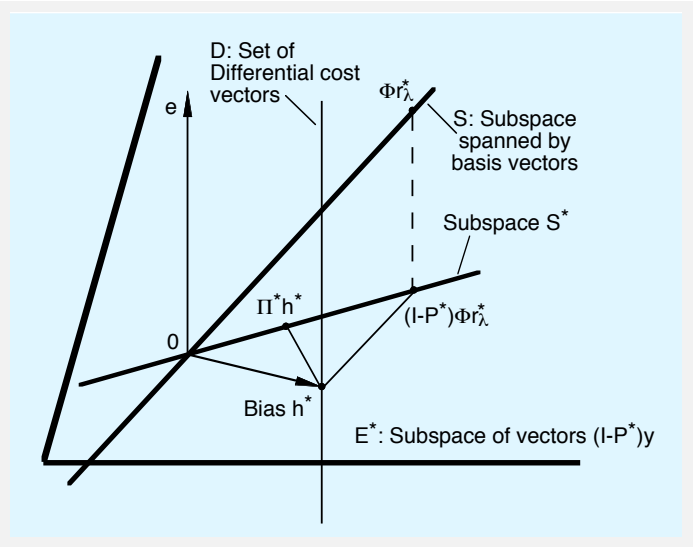

Figure 6.7.1 Illustration of the estimate (6.226). Consider the subspace

$$E^* = \Big\{(I - P^*)y \mid y \in \Re^n\Big\}.$$

Let Ξ be the diagonal matrix with $\xi_1, \ldots, \xi_n$ on the diagonal. Note that:

(a) E^* is the subspace that is orthogonal to the unit vector e in the scaled geometry of the norm $\|\cdot\|_\xi$, in the sense that $e'\Xi z = 0$ for all $z \in E^*$. Indeed we have

$$e'\Xi(I - P^*)y = 0, \qquad \text{for all } y \in \Re^n,$$

because $e'\Xi = \xi'$ and $\xi'(I - P^*) = 0$ as can be easily verified from the fact that the rows of P^* are all equal to ξ'.

(b) Projection onto E^* with respect to the norm $\|\cdot\|_\xi$ is simply multiplication with $(I - P^*)$ (since $P^*y = \xi' ye$, so P^*y is orthogonal to E^* in the scaled geometry of the norm $\|\cdot\|_\xi$). Thus, S^* is the projection of S onto E^*.

(c) We have $h^* \in E^*$ since $(I - P^*)h^* = h^*$ in view of $P^*h^* = 0$.

(d) The equation

$$\min_{h \in D} \|h - \Phi r_\lambda^*\|_\xi = \|h^* - (I - P^*)\Phi r_\lambda^*\|_\xi$$

is geometrically evident from the figure. Also, the term $\|\Pi^* h^* - h^*\|_\xi$ of the error bound is the minimum possible error given that h^* is approximated with an element of S^*.

(e) The estimate (6.226), is the analog of the discounted estimate of Prop. 6.3.5, with E^* playing the role of the entire space, and with the "geometry of the problem" projected onto E^*. Thus, S^* plays the role of S, h^* plays the role of J_μ, $(I - P^*)\Phi r_\lambda^*$ plays the role of Φr_λ^*, and Π^* plays the role of Π. Finally, α_λ is the best possible contraction modulus of $\Pi^* F_{\gamma,\lambda}$ over $\gamma \in (0, 1)$ and within E^* (see the paper [TsV99a] for a detailed analysis).

where γ is a positive stepsize and

$$C_k = \frac{1}{k+1}\sum_{t=0}^{k} z_t\big(\phi(i_t)' - \phi(i_{t+1})'\big), \qquad G_k = \left(\frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\,\phi(i_t)'\right)^{-1},$$

$$d_k = \frac{1}{k+1}\sum_{t=0}^{k} z_t\,\big(g(i_t,i_{t+1}) - \eta_t\big), \qquad z_t = \sum_{m=0}^{t}\lambda^{t-m}\phi(i_m).$$

Scaled versions of this algorithm, where G_k is a scaling matrix are also possible.

The matrices C_k, G_k, and vector d_k can be shown to converge to limits:

$$C_k \to \Phi'\Xi(I - P^{(\lambda)})\Phi, \qquad G_k \to \Phi'\Xi\Phi, \qquad d_k \to \Phi'\Xi\, g^{(\lambda)}, \tag{6.228}$$

where the matrix $P^{(\lambda)}$ is defined by Eq. (6.223), $g^{(\lambda)}$ is given by

$$g^{(\lambda)} = \sum_{\ell=0}^{\infty}\lambda^{\ell}P^{\ell}(g - \eta e),$$

and Ξ is the diagonal matrix with diagonal entries $\xi_1,\ldots,\xi_n$:

$$\Xi = diag(\xi_1,\ldots,\xi_n),$$

[cf. Eqs. (6.85) and (6.86)].

6.7.2 Approximate Policy Iteration

Let us consider an approximate policy iteration method that involves approximate policy evaluation and approximate policy improvement. We assume that *all stationary policies are unichain, and a special state s is recurrent in the Markov chain corresponding to each stationary policy*. As in Section 4.3.1, we consider the stochastic shortest path problem obtained by leaving unchanged all transition probabilities $p_{ij}(u)$ for $j \neq s$, by setting all transition probabilities $p_{is}(u)$ to 0, and by introducing an artificial termination state t to which we move from each state i with probability $p_{is}(u)$. The one-stage cost is equal to $g(i,u) - \eta$, where η is a scalar parameter. We refer to this stochastic shortest path problem as the η-SSP.

The method generates a sequence of stationary policies μ^k, a corresponding sequence of gains η_{μ^k}, and a sequence of cost vectors h_k. We assume that for some $\epsilon > 0$, we have

$$\max_{i=1,\ldots,n}\big|h_k(i) - h_{\mu^k,\eta_k}(i)\big| \leq \epsilon, \qquad k = 0,1,\ldots,$$

where

$$\eta_k = \min_{m=0,1,\ldots,k} \eta_{\mu^m},$$

$h_{\mu^k,\eta_k}(i)$ is the cost-to-go from state i to the reference state s for the η_k-SSP under policy μ^k, and ϵ is a positive scalar quantifying the accuracy of evaluation of the cost-to-go function of the η_k-SSP. Note that we assume *exact* calculation of the gains η_{μ^k}. Note also that we may calculate approximate differential costs $\tilde{h}_k(i,r)$ that depend on a parameter vector r without regard to the reference state s. These differential costs may then be replaced by

$$h_k(i) = \tilde{h}_k(i,r) - \tilde{h}(s,r), \qquad i = 1,\ldots,n.$$

We assume that policy improvement is carried out by approximate minimization in the DP mapping. In particular, we assume that there exists a tolerance $\delta > 0$ such that for all i and k, $\mu^{k+1}(i)$ attains the minimum in the expression

$$\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + h_k(j)\big),$$

within a tolerance $\delta > 0$.

We now note that since η_k is monotonically nonincreasing and is bounded below by the optimal gain η^*, it must converge to some scalar $\overline{\eta}$. Since η_k can take only one of the finite number of values η_μ corresponding to the finite number of stationary policies μ, we see that η_k must converge finitely to $\overline{\eta}$; that is, for some $\overline{k}$, we have

$$\eta_k = \overline{\eta}, \qquad k \geq \overline{k}.$$

Let $h_{\overline{\eta}}(s)$ denote the optimal cost-to-go from state s in the $\overline{\eta}$-SSP. Then, by using Prop. 2.4.1, we have

$$\limsup_{k\to\infty}\big(h_{\mu^k,\overline{\eta}}(s) - h_{\overline{\eta}}(s)\big) \leq \frac{n(1-\rho+n)(\delta+2\epsilon)}{(1-\rho)^2}, \tag{6.229}$$

where

$$\rho = \max_{i=1,\ldots,n,\,\mu} P\big(i_k \neq s,\, k = 1,\ldots,n \mid i_0 = i, \mu\big),$$

and i_k denotes the state of the system after k stages. On the other hand, as can also be seen from Fig. 6.7.2, the relation

$$\overline{\eta} \leq \eta_{\mu^k}$$

implies that

$$h_{\mu^k,\overline{\eta}}(s) \geq h_{\mu^k,\eta_{\mu^k}}(s) = 0.$$

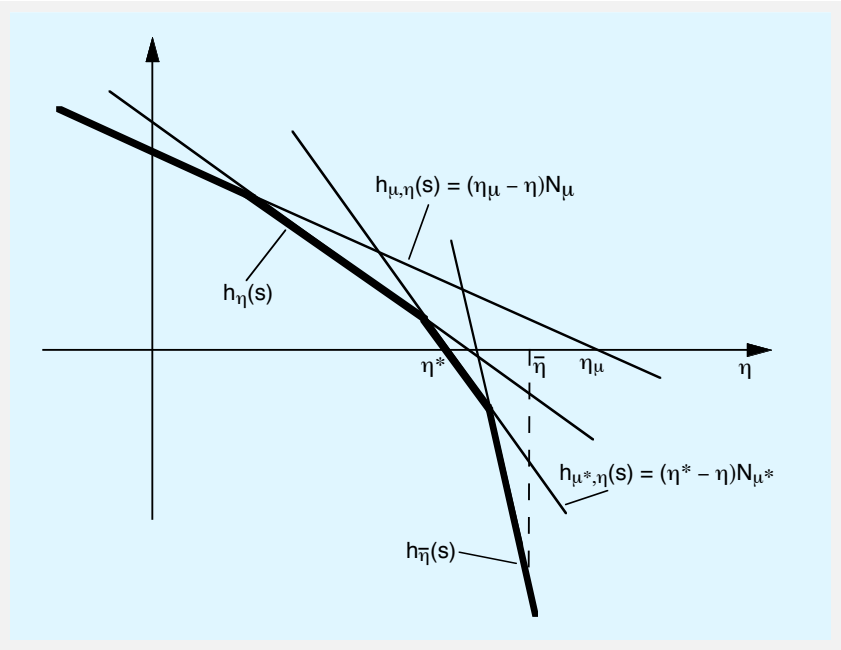

Figure 6.7.2 Relation of the costs of stationary policies for the η-SSP in the approximate policy iteration method. Here, N_μ is the expected number of stages to return to state s, starting from s and using μ. Since $\eta_{\mu^k} \geq \overline{\eta}$, we have

$$h_{\mu^k,\overline{\eta}}(s) \geq h_{\mu^k,\eta_{\mu^k}}(s) = 0.$$

Furthermore, if μ^* is an optimal policy for the η^*-SSP, we have

$$h_{\overline{\eta}}(s) \leq h_{\mu^*,\overline{\eta}}(s) = (\eta^* - \overline{\eta})N_{\mu^*}.$$

It follows, using also Fig. 6.7.2, that

$$h_{\mu^k,\overline{\eta}}(s) - h_{\overline{\eta}}(s) \geq -h_{\overline{\eta}}(s) \geq -h_{\mu^*,\overline{\eta}}(s) = (\overline{\eta} - \eta^*)N_{\mu^*}, \tag{6.230}$$

where μ^* is an optimal policy for the η^*-SSP (and hence also for the original average cost per stage problem) and N_{μ^*} is the expected number of stages to return to state s, starting from s and using μ^*. Thus, from Eqs. (6.229) and (6.230), we have

$$\overline{\eta} - \eta^* \leq \frac{n(1-\rho+n)(\delta+2\epsilon)}{N_{\mu^*}(1-\rho)^2}. \tag{6.231}$$

This relation provides an estimate on the steady-state error of the approximate policy iteration method.

We finally note that optimistic versions of the preceding approximate policy iteration method are harder to implement than their discounted cost counterparts. The reason is our assumption that the gain η_μ of every generated policy μ is exactly calculated; in an optimistic method the current policy μ may not remain constant for sufficiently long time to estimate accurately η_μ. One may consider schemes where an optimistic version of policy iteration is used to solve the η-SSP for a fixed η. The value of η may occasionally be adjusted downward by calculating "exactly" through simulation the gain η_μ of some of the (presumably most promising) generated policies μ, and by then updating η according to $\eta := \min\{\eta, \eta_\mu\}$. An alternative is to approximate the average cost problem with a discounted problem, for which an optimistic version of approximate policy iteration can be readily implemented.

6.7.3 Q-Learning for Average Cost Problems

To derive the appropriate form of the Q-learning algorithm, we form an auxiliary average cost problem by augmenting the original system with one additional state for each possible pair (i, u) with $u \in U(i)$. Thus, the states of the auxiliary problem are those of the original problem, $i = 1, \ldots, n$, together with the additional states (i, u), $i = 1, \ldots, n$, $u \in U(i)$. The probabilistic transition mechanism from an original state i is the same as for the original problem [probability $p_{ij}(u)$ of moving to state j], while the probabilistic transition mechanism from a state (i, u) is that we move only to states j of the original problem with corresponding probabilities $p_{ij}(u)$ and costs $g(i, u, j)$.

It can be seen that the auxiliary problem has the same optimal average cost per stage η as the original, and that the corresponding Bellman's equation is

$$\eta + h(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + h(j)\big), \qquad i = 1, \ldots, n, \quad (6.232)$$

$$\eta + Q(i, u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + h(j)\big), \qquad i = 1, \ldots, n, \ u \in U(i), \quad (6.233)$$

where $Q(i, u)$ is the differential cost corresponding to (i, u). Taking the minimum over u in Eq. (6.233) and comparing with Eq. (6.232), we obtain

$$h(i) = \min_{u \in U(i)} Q(i, u), \qquad i = 1, \ldots, n.$$

Substituting the above form of $h(i)$ in Eq. (6.233), we obtain Bellman's equation in a form that exclusively involves the Q-factors:

$$\eta + Q(i, u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i, u, j) + \min_{v \in U(j)} Q(j, v) \right), \ i = 1, \ldots, n, \ u \in U(i).$$

Let us now apply to the auxiliary problem the following variant of the relative value iteration

$$h^{k+1} = Th^k - h^k(s)e,$$

where s is a special state. We then obtain the iteration [cf. Eqs. (6.232) and (6.233)]

$$h^{k+1}(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + h^k(j)\big) - h^k(s), \qquad i = 1,\ldots,n,$$

$$Q^{k+1}(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)+h^k(j)\big)-h^k(s), \qquad i = 1,\ldots,n,\ u \in U(i). \tag{6.234}$$

From these equations, we have that

$$h^k(i) = \min_{u\in U(i)} Q^k(i,u), \qquad i = 1,\ldots,n,$$

and by substituting the above form of h^k in Eq. (6.234), we obtain the following relative value iteration for the Q-factors

$$Q^{k+1}(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \min_{v\in U(j)} Q^k(j,v)\right) - \min_{v\in U(s)} Q^k(s,v).$$

The sequence of values $\min_{u\in U(s)} Q^k(s,u)$ is expected to converge to the optimal average cost per stage and the sequences of values $\min_{u\in U(i)} Q(i,u)$ are expected to converge to differential costs $h(i)$.

An incremental version of the preceding iteration that involves a positive stepsize γ is given by

$$Q(i,u) := (1-\gamma)Q(i,u) + \gamma\left(\sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \min_{v\in U(j)} Q(j,v)\right) - \min_{v\in U(s)} Q(s,v)\right).$$

The natural form of the Q-learning method for the average cost problem is an approximate version of this iteration, whereby the expected value is replaced by a single sample, i.e.,

$$Q(i,u) := Q(i,u) + \gamma\Big(g(i,u,j) + \min_{v\in U(j)} Q(j,v) - \min_{v\in U(s)} Q(s,v) - Q(i,u)\Big),$$

where j and $g(i,u,j)$ are generated from the pair (i,u) by simulation. In this method, only the Q-factor corresponding to the currently sampled pair (i,u) is updated at each iteration, while the remaining Q-factors remain unchanged. Also the stepsize should be diminishing to 0. A convergence analysis of this method can be found in the paper by Abounadi, Bertsekas, and Borkar [ABB01].

Q-Learning Based on the Contracting Value Iteration

We now consider an alternative Q-learning method, which is based on the contracting value iteration method of Section 4.3. If we apply this method to the auxiliary problem used above, we obtain the following algorithm

$$h^{k+1}(i) = \min_{u \in U(i)} \left[\sum_{j=1}^{n} p_{ij}(u) g(i,u,j) + \sum_{\substack{j=1 \\ j \neq s}}^{n} p_{ij}(u) h^k(j) \right] - \eta^k, \quad (6.235)$$

$$Q^{k+1}(i,u) = \sum_{j=1}^{n} p_{ij}(u) g(i,u,j) + \sum_{\substack{j=1 \\ j \neq s}}^{n} p_{ij}(u) h^k(j) - \eta^k, \quad (6.236)$$

$$\eta^{k+1} = \eta^k + \alpha^k h^{k+1}(s).$$

From these equations, we have that

$$h^k(i) = \min_{u \in U(i)} Q^k(i,u),$$

and by substituting the above form of h^k in Eq. (6.236), we obtain

$$Q^{k+1}(i,u) = \sum_{j=1}^{n} p_{ij}(u) g(i,u,j) + \sum_{\substack{j=1 \\ j \neq s}}^{n} p_{ij}(u) \min_{v \in U(j)} Q^k(j,v) - \eta^k,$$

$$\eta^{k+1} = \eta^k + \alpha^k \min_{v \in U(s)} Q^{k+1}(s,v).$$

A small-stepsize version of this iteration is given by

$$Q(i,u) := (1-\gamma) Q(i,u) + \gamma \left(\sum_{j=1}^{n} p_{ij}(u) g(i,u,j) + \sum_{\substack{j=1 \\ j \neq s}}^{n} p_{ij}(u) \min_{v \in U(j)} Q(j,v) - \eta \right),$$

$$\eta := \eta + \alpha \min_{v \in U(s)} Q(s,v),$$

where γ and α are positive stepsizes. A natural form of Q-learning based on this iteration is obtained by replacing the expected values by a single sample, i.e.,

$$Q(i,u) := (1-\gamma) Q(i,u) + \gamma \Big(g(i,u,j) + \min_{v \in U(j)} \hat{Q}(j,v) - \eta \Big), \quad (6.237)$$

$$\eta := \eta + \alpha \min_{v \in U(s)} Q(s,v), \tag{6.238}$$

where

$$\hat{Q}(j,v) = \begin{cases} Q(j,v) & \text{if } j \neq s, \\ 0 & \text{otherwise,} \end{cases}$$

and j and $g(i,u,j)$ are generated from the pair (i,u) by simulation. Here the stepsizes γ and α should be diminishing, but α should diminish "faster" than γ; i.e., the ratio of the stepsizes α/γ should converge to zero. For example, we may use $\gamma = C/k$ and $\alpha = c/k \log k$, where C and c are positive constants and k is the number of iterations performed on the corresponding pair (i,u) or η, respectively.

The algorithm has two components: the iteration (6.237), which is essentially a Q-learning method that aims to solve the η-SSP for the current value of η, and the iteration (6.238), which updates η towards its correct value η^*. However, η is updated at a slower rate than Q, since the stepsize ratio α/γ converges to zero. The effect is that the Q-learning iteration (6.237) is fast enough to keep pace with the slower changing η-SSP. A convergence analysis of this method can also be found in the paper [ABB01].

6.8 SIMULATION-BASED SOLUTION OF LARGE SYSTEMS

We have focused so far in this chapter on approximating the solution of Bellman equations within a subspace of basis functions in a variety of contexts. We have seen common analytical threads across discounted, SSP, and average cost problems, as well as differences in formulations, implementation details, and associated theoretical results. In this section we will aim for a more general view of simulation-based solution of large systems within which the methods and analysis so far can be understood and extended. The benefit of this analysis is a deeper perspective, and the ability to address more general as well as new problems in DP and beyond.

For most of this section we consider simulation-based methods for solving the linear fixed point equation

$$x = b + Ax,$$

where A is an $n \times n$ matrix and b is an n-dimensional vector, with components denoted a_{ij} and b_i, respectively. These methods are divided in two major categories, which are based on distinctly different philosophies and lines of analysis:

(a) *Stochastic approximation methods*, which have the form

$$x_{k+1} = (1 - \gamma_k)x_k + \gamma_k(b + Ax_k + w_k),$$

where w_k is zero-mean noise. Here the term $b + Ax_k + w_k$ may be viewed as a simulation sample of $b + Ax$, and γ_k is a diminishing positive stepsize ($\gamma_k \downarrow 0$). These methods were discussed briefly in Section 6.1.6. A prime example within our context is TD(λ), which is a stochastic approximation method for solving the (linear) multistep projected equation $C^{(\lambda)}r = d^{(\lambda)}$ corresponding to evaluation of a single policy (cf. Section 6.3.6). The Q-learning algorithm of Section 6.5.1 is also a stochastic approximation method, but it solves a nonlinear fixed point problem - Bellman's equation for multiple policies.

(b) *Monte-Carlo estimation methods*, which obtain Monte-Carlo estimates A_m and b_m, based on m samples, and use them in place of A and b in various deterministic methods. Thus an approximate fixed point may be obtained by matrix inversion,

$$\hat{x} = (I - A_m)^{-1}b_m,$$

or iteratively by

$$x_{k+1} = (1 - \gamma)x_k + \gamma(b_m + A_m x_k), \qquad k = 0, 1, \ldots, \tag{6.239}$$

where γ is a constant positive stepsize. In a variant of the iterative approach the estimates A_m and b_m are updated as the simulation samples are collected, in which case the method (6.239) takes the form

$$x_{k+1} = (1 - \gamma)x_k + \gamma(b_k + A_k x_k), \qquad k = 0, 1, \ldots.$$

The LSTD-type methods are examples of the matrix inversion approach, while the LSPE-type methods are examples of the iterative approach.

Stochastic approximation methods, generally speaking, tend to be simpler but slower. They are simpler partly because they involve a single vector sample rather than matrix-vector estimates that are based on many samples. They are slower because their iterations involve more noise per iteration (a single sample rather than a Monte-Carlo average), and hence require a diminishing stepsize. Basically, stochastic approximation methods combine the iteration and Monte-Carlo estimation processes, while methods such as Eq. (6.239) separate the two processes to a large extent.

We should also mention that the fixed point problem $x = b + Ax$ may involve approximations or multistep mappings (cf. Section 6.3.6). For example it may result from a projected equation approach or from an aggregation approach.

In this section, we will focus on Monte-Carlo estimation methods where x is approximated within a subspace $S = \{\Phi r \mid r \in \Re^s\}$. In the special case where $\Phi = I$, we obtain lookup table-type methods, where there

is no subspace approximation. We start with the projected equation approach, we continue with the related Bellman equation error methods, and finally we consider aggregation approaches. On occasion we discuss various extensions, involving for example nonlinear fixed point problems. We do not provide a rigorous discussion of stochastic approximation methods, as this would require the use of mathematics that are beyond our scope. We refer to the extensive literature on the subject (see the discussion of Section 6.1.6).

6.8.1 Projected Equations - Simulation-Based Versions

We first focus on general linear fixed point equations $x = T(x)$, where

$$T(x) = b + Ax, \tag{6.240}$$

A is an $n \times n$ matrix, and $b \in \Re^n$ is a vector. We consider approximations of a solution by solving a projected equation

$$\Phi r = \Pi T(\Phi r) = \Pi(b + A\Phi r),$$

where Π denotes projection with respect to a weighted Euclidean norm $\|\cdot\|_\xi$ on a subspace

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

We assume throughout that the columns of the $n \times s$ matrix Φ are linearly independent basis functions.

Examples are Bellman's equation for policy evaluation, in which case $A = \alpha P$, where P is a transition matrix (discounted and average cost), or P is a substochastic matrix (row sums less than or equal to 0, as in SSP), and $\alpha = 1$ (SSP and average cost), or $\alpha < 1$ (discounted). Other examples in DP include the semi-Markov problems discussed in Chapter 5. However, for the moment we do not assume the presence of any stochastic structure in A. Instead, *we assume throughout that $I - \Pi A$ is invertible*, so that the projected equation has a unique solution denoted r^*.

We will derive extensions of LSTD(0), LSPE(0), and TD(0) methods of Section 6.3 (the latter two under the assumption that ΠT is a contraction). References [BeY07] and [BeY09], where these methods were first developed, provide extensions of LSTD(λ), LSPE(λ), and TD(λ) for $\lambda \in (0, 1)$; the later two are convergent when $\Pi T^{(\lambda)}$ is a contraction on S, where

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

and T has the general form $T(x) = b + Ax$ of Eq. (6.240) (cf. Section 6.3.6).

Even if T or ΠT are not contractions, we can obtain an error bound that generalizes some of the bounds obtained earlier. We have

$$x^* - \Phi r^* = x^* - \Pi x^* + \Pi T x^* - \Pi T \Phi r^* = x^* - \Pi x^* + \Pi A(x^* - \Phi r^*), \tag{6.241}$$

from which

$$x^* - \Phi r^* = (I - \Pi A)^{-1}(x^* - \Pi x^*).$$

Thus, for any norm $\|\cdot\|$ and fixed point x^* of T,

$$\|x^* - \Phi r^*\| \leq \big\|(I - \Pi A)^{-1}\big\| \, \|x^* - \Pi x^*\|, \tag{6.242}$$

so the approximation error $\|x^* - \Phi r^*\|$ is proportional to the distance of the solution x^* from the approximation subspace. If ΠT is a contraction mapping of modulus $\alpha \in (0,1)$ with respect to $\|\cdot\|$, from Eq. (6.241), we have

$$\|x^*-\Phi r^*\| \leq \|x^*-\Pi x^*\|+\|\Pi T(x^*)-\Pi T(\Phi r^*)\| \leq \|x^*-\Pi x^*\|+\alpha\|x^*-\Phi r^*\|,$$

so that

$$\|x^* - \Phi r^*\| \leq \frac{1}{1-\alpha}\|x^* - \Pi x^*\|. \tag{6.243}$$

We first introduce an equivalent form of the projected equation $\Phi r = \Pi(b + A\Phi r)$, which generalizes the matrix form (6.40)-(6.41) for discounted DP problems. Let us assume that the positive probability distribution vector ξ is given. By the definition of projection with respect to $\|\cdot\|_\xi$, the unique solution r^* of this equation satisfies

$$r^* = \arg\min_{r \in \Re^s} \big\|\Phi r - (b + A\Phi r^*)\big\|_\xi^2.$$

Setting to 0 the gradient with respect to r, we obtain the corresponding orthogonality condition

$$\Phi'\Xi\big(\Phi r^* - (b + A\Phi r^*)\big) = 0,$$

where Ξ is the diagonal matrix with the probabilities $\xi_1, \ldots, \xi_n$ along the diagonal. Equivalently,

$$Cr^* = d,$$

where

$$C = \Phi'\Xi(I - A)\Phi, \qquad d = \Phi'\Xi b, \tag{6.244}$$

and Ξ is the diagonal matrix with the components of ξ along the diagonal [cf. Eqs. (6.40)-(6.41)].

We will now develop a simulation-based approximation to the system $Cr^* = d$, by using corresponding estimates of C and d. We write C and d as expected values with respect to ξ:

$$C = \sum_{i=1}^n \xi_i \phi(i) \left(\phi(i) - \sum_{j=1}^n a_{ij}\phi(j)\right)', \qquad d = \sum_{i=1}^n \xi_i \phi(i) b_i. \tag{6.245}$$

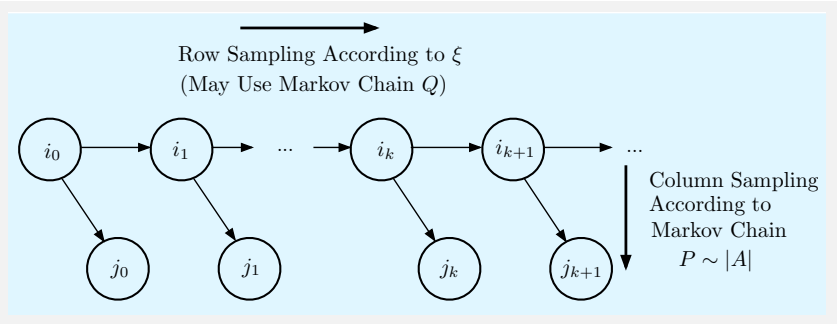

Figure 6.8.1 The basic simulation methodology consists of (a) generating a sequence of indices $\{i_0, i_1, \ldots\}$ according to the distribution ξ (a Markov chain Q may be used for this, but this is not a requirement), and (b) generating a sequence of transitions $\big\{(i_0, j_0), (i_1, j_1), \ldots\big\}$ using a Markov chain P. It is possible that $j_k = i_{k+1}$, but this is not necessary.

As in Section 6.3.3, we approximate these expected values by simulation-obtained sample averages, however, here we do not have a Markov chain structure by which to generate samples. We must therefore design a sampling scheme that can be used to properly approximate the expected values in Eq. (6.245). In the most basic form of such a scheme, we generate a sequence of indices $\{i_0, i_1, \ldots\}$, and a sequence of transitions between indices $\big\{(i_0, j_0), (i_1, j_1), \ldots\big\}$. We use any probabilistic mechanism for this, subject to the following two requirements (cf. Fig. 6.8.1):

(1) **Row sampling:** The sequence $\{i_0, i_1, \ldots\}$ is generated according to the distribution ξ, which defines the projection norm $\|\cdot\|_\xi$, in the sense that with probability 1,

$$\lim_{k\to\infty} \frac{\sum_{t=0}^{k} \delta(i_t = i)}{k+1} = \xi_i, \qquad i = 1, \ldots, n, \tag{6.246}$$

where $\delta(\cdot)$ denotes the indicator function [$\delta(E) = 1$ if the event E has occurred and $\delta(E) = 0$ otherwise].

(2) **Column sampling:** The sequence $\big\{(i_0, j_0), (i_1, j_1), \ldots\big\}$ is generated according to a certain stochastic matrix P with transition probabilities p_{ij} that satisfy

$$p_{ij} > 0 \qquad \text{if} \qquad a_{ij} \neq 0, \tag{6.247}$$

in the sense that with probability 1,

$$\lim_{k\to\infty} \frac{\sum_{t=0}^{k} \delta(i_t = i, j_t = j)}{\sum_{t=0}^{k} \delta(i_t = i)} = p_{ij}, \qquad i, j = 1, \ldots, n. \tag{6.248}$$

At time k, we approximate C and d with

$$C_k = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\left(\phi(i_t) - \frac{a_{i_t j_t}}{p_{i_t j_t}}\phi(j_t)\right)', \quad d_k = \frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)b_{i_t}. \tag{6.249}$$

To show that this is a valid approximation, similar to the analysis of Section 6.3.3, we count the number of times an index occurs and after collecting terms, we write Eq. (6.249) as

$$C_k = \sum_{i=1}^{n}\hat{\xi}_{i,k}\phi(i)\left(\phi(i) - \sum_{j=1}^{n}\hat{p}_{ij,k}\frac{a_{ij}}{p_{ij}}\phi(j)\right)', \qquad d_k = \sum_{i=1}^{n}\hat{\xi}_{i,k}\phi(i)b_i, \tag{6.250}$$

where

$$\hat{\xi}_{i,k} = \frac{\sum_{t=0}^{k}\delta(i_t = i)}{k+1}, \qquad \hat{p}_{ij,k} = \frac{\sum_{t=0}^{k}\delta(i_t = i, j_t = j)}{\sum_{t=0}^{k}\delta(i_t = i)};$$

(cf. the calculations in Section 6.3.3). In view of the assumption

$$\hat{\xi}_{i,k} \to \xi_i, \qquad \hat{p}_{ij,k} \to p_{ij}, \qquad i,j = 1,\ldots,n,$$

[cf. Eqs. (6.246) and (6.248)], by comparing Eqs. (6.245) and (6.250), we see that $C_k \to C$ and $d_k \to d$. Since the solution r^* of the system (6.245) exists and is unique, the same is true for the system (6.250) for all t sufficiently large. Thus, with probability 1, the solution of the system (6.249) converges to r^* as $k \to \infty$.

A comparison of Eqs. (6.245) and (6.250) indicates some considerations for selecting the stochastic matrix P. It can be seen that "important" (e.g., large) components a_{ij} should be simulated more often (p_{ij}: large). In particular, if (i,j) is such that $a_{ij} = 0$, there is an incentive to choose $p_{ij} = 0$, since corresponding transitions (i,j) are "wasted" in that they do not contribute to improvement of the approximation of Eq. (6.245) by Eq. (6.250). This suggests that the structure of P should match in some sense the structure of the matrix A, to improve the efficiency of the simulation (the number of samples needed for a given level of simulation error variance). On the other hand, the choice of P does not affect the limit of $\Phi\hat{r}_k$, which is the solution Φr^* of the projected equation. By contrast, the choice of ξ affects the projection Π and hence also Φr^*.

Note that there is a lot of flexibility for generating the sequence $\{i_0, i_1, \ldots\}$ and the transition sequence $\{(i_0,j_0),(i_1,j_1),\ldots\}$ to satisfy Eqs. (6.246) and (6.248). For example, to satisfy Eq. (6.246), the indices i_t do not need to be sampled independently according to ξ. Instead, it may be convenient to introduce an irreducible Markov chain with transition matrix Q, states $1,\ldots,n$, and ξ as its steady-state probability vector, and to start

at some state i_0 and generate the sequence $\{i_0, i_1, \ldots\}$ as a single infinitely long trajectory of the chain. For the transition sequence, we may optionally let $j_k = i_{k+1}$ for all k, in which case P would be identical to Q, but in general this is not essential.

Let us discuss two possibilities for constructing a Markov chain with steady-state probability vector ξ. The first is useful when a desirable distribution ξ is known up to a normalization constant. Then we can construct such a chain using techniques that are common in Markov chain Monte Carlo (MCMC) methods (see e.g., Liu [Liu01], Rubinstein and Kroese [RuK08]).

The other possibility, which is useful when there is no particularly desirable ξ, is to specify first the transition matrix Q of the Markov chain and let ξ be its steady-state probability vector. Then the requirement (6.246) will be satisfied if the Markov chain is irreducible, in which case ξ will be the unique steady-state probability vector of the chain and will have positive components. An important observation is that explicit knowledge of ξ is not required; it is just necessary to know the Markov chain and to be able to simulate its transitions. The approximate DP applications of Sections 6.3, 6.6, and 6.7, where $Q = P$, fall into this context. In the next section, we will discuss favorable methods for constructing the transition matrix Q from A, which result in ΠT being a contraction so that iterative methods are applicable.

Note that multiple simulated sequences can be used to form the equation (6.249). For example, in the Markov chain-based sampling schemes, we can generate multiple infinitely long trajectories of the chain, starting at several different states, and for each trajectory use $j_k = i_{k+1}$ for all k. This will work even if the chain has multiple recurrent classes, as long as there are no transient states and at least one trajectory is started from within each recurrent class. Again ξ will be a steady-state probability vector of the chain, and need not be known explicitly. Note also that using multiple trajectories may be interesting even if there is a single recurrent class, for at least two reasons:

(a) The generation of trajectories may be parallelized among multiple processors, resulting in significant speedup.

(b) The empirical frequencies of occurrence of the states may approach the steady-state probabilities more quickly; this is particularly so for large and "stiff" Markov chains.

We finally note that the option of using distinct Markov chains Q and P for row and column sampling is important in the DP/policy iteration context. In particular, by using a distribution ξ that is not associated with P, we may resolve the issue of exploration (see Section 6.3.7).

6.8.2 Matrix Inversion and Regression-Type Methods

Given simulation-based estimates C_k and d_k of C and d, respectively, we may approximate $r^* = C^{-1}d$ with

$$\hat{r}_k = C_k^{-1} d_k,$$

in which case we have $\hat{r}_k \to r^*$ with probability 1 (this parallels the LSTD method of Section 6.3.4). An alternative, which is more suitable for the case where C_k is nearly singular, is the regression/regularization-based estimate

$$\hat{r}_k = (C_k' \Sigma^{-1} C_k + \beta I)^{-1} (C_k' \Sigma^{-1} d_k + \beta \bar{r}), \tag{6.251}$$

[cf. Eq. (6.58) in Section 6.3.4], where $\bar{r}$ is an *a priori* estimate of $r^* = C^{-1}d$, β is a positive scalar, and Σ is some positive definite symmetric matrix. The error estimate given by Prop. 6.3.4 applies to this method. In particular, the error $\|\hat{r}_k - r^*\|$ is bounded by the sum of two terms: one due to simulation error (which is larger when C is nearly singular, and decreases with the amount of sampling used), and the other due to regularization error (which depends on the regularization parameter β and the error $\|\bar{r} - r^*\|$); cf. Eq. (6.60).

To obtain a confidence interval for the error $\|\hat{r}_k - r^*\|$, we view all variables generated by simulation to be random variables on a common probability space. Let $\hat{\Sigma}_k$ be the covariance of $(d_k - C_k r^*)$, and let

$$\hat{b}_k = \hat{\Sigma}_k^{-1/2} (d_k - C_k r^*).$$

Note that $\hat{b}_k$ has covariance equal to the identity. Let $\hat{P}_k$ be the cumulative distribution function of $\|\hat{b}_k\|^2$, and note that

$$\|\hat{b}_k\| \leq \sqrt{\hat{P}_k^{-1}(1-\theta)} \tag{6.252}$$

with probability $(1-\theta)$, where $\hat{P}_k^{-1}(1-\theta)$ is the threshold value v at which the probability that $\|\hat{b}_k\|^2$ takes value greater than v is θ. We denote by $\mathbf{P}(E)$ the probability of an event E.

Proposition 6.8.1: We have

$$\mathbf{P}\Big(\|\hat{r}_k - r^*\| \leq \sigma_k(\Sigma,\beta)\Big) \geq 1-\theta,$$

where

$$\begin{aligned}\sigma_k(\Sigma,\beta) = &\max_{i=1,\ldots,s}\left\{\frac{\lambda_i}{\lambda_i^2+\beta}\right\}\left\|\Sigma^{-1/2}\hat{\Sigma}_k^{1/2}\right\|\sqrt{\hat{P}_k^{-1}(1-\theta)}\\ &+\max_{i=1,\ldots,s}\left\{\frac{\beta}{\lambda_i^2+\beta}\right\}\|\bar{r}-r^*\|,\end{aligned} \tag{6.253}$$

and $\lambda_1,\ldots,\lambda_s$ are the singular values of $\Sigma^{-1/2}C_k$.

Proof: Let $b_k = \Sigma^{-1/2}(d_k - C_k r^*)$. Following the notation and proof of Prop. 6.3.4, and using the relation $\hat{b}_k = \hat{\Sigma}_k^{-1/2}\Sigma^{1/2}b_k$, we have

$$\begin{aligned}\hat{r}_k - r^* &= V(\Lambda^2+\beta I)^{-1}\Lambda U' b_k + \beta\, V(\Lambda^2+\beta I)^{-1}V'(\bar{r}-r^*)\\ &= V(\Lambda^2+\beta I)^{-1}\Lambda U'\,\Sigma^{-1/2}\hat{\Sigma}_k^{1/2}\hat{b}_k + \beta\, V(\Lambda^2+\beta I)^{-1}V'(\bar{r}-r^*).\end{aligned}$$

From this, we similarly obtain

$$\|\hat{r}_k - r^*\| \leq \max_{i=1,\ldots,s}\left\{\frac{\lambda_i}{\lambda_i^2+\beta}\right\}\left\|\Sigma^{-1/2}\hat{\Sigma}_k^{1/2}\right\|\|\hat{b}_k\| + \max_{i=1,\ldots,s}\left\{\frac{\beta}{\lambda_i^2+\beta}\right\}\|\bar{r}-r^*\|.$$

Since Eq. (6.252) holds with probability $(1-\theta)$, the desired result follows. **Q.E.D.**

Using a form of the central limit theorem, we may assume that for a large number of samples, $\hat{b}_k$ asymptotically becomes a Gaussian random s-dimensional vector, so that the random variable

$$\|\hat{b}_k\|^2 = (d_k - C_k r^*)'\hat{\Sigma}_k^{-1}(d_k - C_k r^*)$$

can be treated as a chi-square random variable with s degrees of freedom (since the covariance of $\hat{b}_k$ is the identity by definition). Assuming this, the distribution $\hat{P}_k^{-1}(1-\theta)$ in Eq. (6.253) is approximately equal and may be replaced by $P^{-1}(1-\theta;s)$, the threshold value v at which the probability that a chi-square random variable with s degrees of freedom takes value greater than v is θ. Thus in a practical application of Prop. 6.8.1, one may replace $\hat{P}_k^{-1}(1-\theta)$ by $P^{-1}(1-\theta;s)$, and also replace $\hat{\Sigma}_k$ with an estimate of the covariance of $(d_k - C_k r^*)$; the other quantities in Eq. (6.253) (Σ, λ_i, β, and $\overline{r}$) are known.

6.8.3 Iterative/LSPE-Type Methods

In this section, we will consider iterative methods for solving the projected equation $Cr = d$ [cf. Eq. (6.245)], using simulation-based estimates C_k and d_k. We first consider the fixed point iteration

$$\Phi r_{k+1} = \Pi T(\Phi r_k), \qquad k = 0, 1, \ldots, \tag{6.254}$$

which generalizes the PVI method of Section 6.3.2. For this method to be valid and to converge to r^* it is essential that ΠT is a contraction with respect to some norm. In the next section, we will provide tools for verifying that this is so.

Similar to the analysis of Section 6.3.3, the simulation-based approximation (LSPE analog) is

$$r_{k+1} = \left(\sum_{t=0}^{k} \phi(i_t)\phi(i_t)'\right)^{-1} \sum_{t=0}^{k} \phi(i_t) \left(\frac{a_{i_t j_t}}{p_{i_t j_t}} \phi(j_t)' r_k + b_{i_t}\right). \tag{6.255}$$

Here again $\{i_0, i_1, \ldots\}$ is an index sequence and $\{(i_0, j_0), (i_1, j_1), \ldots\}$ is a transition sequence satisfying Eqs. (6.246)-(6.248).

A generalization of this iteration, written in more compact form and introducing scaling with a matrix G_k, is given by

$$r_{k+1} = r_k - \gamma G_k(C_k r_k - d_k), \tag{6.256}$$

where C_k and d_k are given by Eq. (6.249) [cf. Eq. (6.71)]. As in Section 6.3.4, this iteration can be equivalently written in terms of generalized temporal differences as

$$r_{k+1} = r_k - \frac{\gamma}{k+1} G_k \sum_{t=0}^{k} \phi(i_t) q_{k,t}$$

where

$$q_{k,t} = \phi(i_t)' r_k - \frac{a_{i_t j_t}}{p_{i_t j_t}} \phi(j_t)' r_k - b_{i_t}$$

[cf. Eq. (6.72)]. The scaling matrix G_k should converge to an appropriate matrix G.

For the scaled LSPE-type method (6.256) to converge to r^*, we must have $G_k \to G$, $C_k \to C$, and G, C, and γ must be such that $I - \gamma GC$ is a contraction. Noteworthy special cases where this is so are:

(a) The case of iteration (6.255), where $\gamma = 1$ and

$$G_k = \left(\sum_{t=0}^{k} \phi(i_t)\phi(i_t)'\right)^{-1},$$

under the assumption that ΠT is a contraction. The reason is that this iteration asymptotically becomes the fixed point iteration $\Phi r_{k+1} = \Pi T(\Phi r_k)$ [cf. Eq. (6.254)].

(b) C is positive definite, G is symmetric positive definite, and γ is sufficiently small. This case arises in various DP contexts, e.g., the discounted problem where $A = \alpha P$ (cf. Section 6.3).

(c) C is positive definite, $\gamma = 1$, and G has the form

$$G = (C + \beta I)^{-1},$$

where β is a positive scalar (cf. Section 6.3.4). The corresponding iteration (6.256) takes the form

$$r_{k+1} = r_k - (C_k + \beta I)^{-1}(C_k r_k - d_k)$$

[cf. Eq. (6.77)].

(d) C is invertible, $\gamma = 1$, and G has the form

$$G = (C'\Sigma^{-1}C + \beta I)^{-1}C'\Sigma^{-1},$$

where Σ is some positive definite symmetric matrix, and β is a positive scalar. The corresponding iteration (6.256) takes the form

$$r_{k+1} = (C_k'\Sigma_k^{-1}C_k + \beta I)^{-1}(C_k'\Sigma_k^{-1}d_k + \beta r_k)$$

[cf. Eq. (6.76)]. As shown in Section 6.3.2, the eigenvalues of GC are $\lambda_i/(\lambda_i + \beta)$, where λ_i are the eigenvalues of $C'\Sigma^{-1}C$, so $I - GC$ has real eigenvalues in the interval $(0, 1)$. This iteration also works if C is not invertible.

The Analog of TD(0)

Let us also note the analog of the TD(0) method. It is similar to Eq. (6.256), but uses only the last sample:

$$r_{k+1} = r_k - \gamma_k \phi(i_k) q_{k,k},$$

where the stepsize γ_k must be diminishing to 0. It was shown in [BeY07] and [BeY09] that if ΠT is a contraction on S with respect to $\|\cdot\|_\xi$, then the matrix C of Eq. (6.244) is negative definite, which is what is essentially needed for convergence of the method to the solution of the projected equation $Cr = d$.

Contraction Properties

We will now derive conditions for ΠT to be a contraction, which facilitates the use of the preceding iterative methods. We assume that the index sequence $\{i_0, i_1, \ldots\}$ is generated as an infinitely long trajectory of a Markov chain whose steady-state probability vector is ξ. We denote by Q the corresponding transition probability matrix and by q_{ij} the components of Q. As discussed earlier, Q may not be the same as P, which is used to generate the transition sequence $\big\{(i_0, j_0), (i_1, j_1), \ldots\big\}$ to satisfy Eqs. (6.246) and (6.248). It seems hard to guarantee that ΠT is a contraction mapping, unless $|A| \leq Q$ [i.e., $|a_{ij}| \leq q_{ij}$ for all (i, j)]. The following propositions assume this condition.

Proposition 6.8.2: Assume that Q is irreducible and that $|A| \leq Q$. Then T and ΠT are contraction mappings under any one of the following three conditions:

(1) For some scalar $\alpha \in (0, 1)$, we have $|A| \leq \alpha Q$.

(2) There exists an index $\overline{i}$ such that $|a_{\overline{i}j}| < q_{\overline{i}j}$ for all $j = 1, \ldots, n$.

(3) There exists an index $\overline{i}$ such that $\sum_{j=1}^{n} |a_{\overline{i}j}| < 1$.

Proof: For any vector or matrix X, we denote by $|X|$ the vector or matrix that has as components the absolute values of the corresponding components of X. Let ξ be the steady-state probability vector of Q. Assume condition (1). Since Π is nonexpansive with respect to $\|\cdot\|_\xi$, it will suffice to show that A is a contraction with respect to $\|\cdot\|_\xi$. We have

$$|Az| \leq |A|\,|z| \leq \alpha Q|z|, \qquad \forall\ z \in \Re^n. \tag{6.257}$$

Using this relation, we obtain

$$\|Az\|_\xi \leq \alpha \|Q|z|\|_\xi \leq \alpha \|z\|_\xi, \qquad \forall\ z \in \Re^n, \tag{6.258}$$

where the last inequality follows since $\|Qx\|_\xi \leq \|x\|_\xi$ for all $x \in \Re^n$ (see Lemma 6.3.1). Thus, A is a contraction with respect to $\|\cdot\|_\xi$ with modulus α.

Assume condition (2). Then, in place of Eq. (6.257), we have

$$|Az| \leq |A|\,|z| \leq Q|z|, \qquad \forall\ z \in \Re^n,$$

with strict inequality for the row corresponding to $\overline{i}$ when $z \neq 0$, and in place of Eq. (6.258), we obtain

$$\|Az\|_\xi < \|Q|z|\|_\xi \leq \|z\|_\xi, \qquad \forall\ z \neq 0.$$

It follows that A is a contraction with respect to $\|\cdot\|_\xi$, with modulus $\max_{\|z\|_\xi \leq 1} \|Az\|_\xi$.

Assume condition (3). It will suffice to show that the eigenvalues of ΠA lie strictly within the unit circle.† Let $\bar{Q}$ be the matrix which is identical to Q except for the $\overline{i}$th row which is identical to the $\overline{i}$th row of $|A|$. From the irreducibility of Q, it follows that for any $i_1 \neq \overline{i}$ it is possible to find a sequence of nonzero components $\bar{Q}_{i_1 i_2}, \ldots, \bar{Q}_{i_{k-1} i_k}, \bar{Q}_{i_k \overline{i}}$ that "lead" from i_1 to $\overline{i}$. Using a well-known result, we have $\bar{Q}^t \to 0$. Since $|A| \leq \bar{Q}$, we also have $|A|^t \to 0$, and hence also $A^t \to 0$ (since $|A^t| \leq |A|^t$). Thus, all eigenvalues of A are strictly within the unit circle. We next observe that from the proof argument under conditions (1) and (2), we have

$$\|\Pi A z\|_\xi \leq \|z\|_\xi, \qquad \forall\ z \in \Re^n,$$

so the eigenvalues of ΠA cannot lie outside the unit circle.

Assume to arrive at a contradiction that ν is an eigenvalue of ΠA with $|\nu| = 1$, and let ζ be a corresponding eigenvector. We claim that $A\zeta$ must have both real and imaginary components in the subspace S. If this were not so, we would have $A\zeta \neq \Pi A\zeta$, so that

$$\|A\zeta\|_\xi > \|\Pi A\zeta\|_\xi = \|\nu\zeta\|_\xi = |\nu|\,\|\zeta\|_\xi = \|\zeta\|_\xi,$$

which contradicts the fact $\|Az\|_\xi \leq \|z\|_\xi$ for all z, shown earlier. Thus, the real and imaginary components of $A\zeta$ are in S, which implies that $A\zeta = \Pi A\zeta = \nu\zeta$, so that ν is an eigenvalue of A. This is a contradiction because $|\nu| = 1$, while the eigenvalues of A are strictly within the unit circle. **Q.E.D.**

Note that the preceding proof has shown that under conditions (1) and (2) of Prop. 6.8.2, T and ΠT are contraction mappings with respect to the specific norm $\|\cdot\|_\xi$, and that under condition (1), the modulus of contraction is α. Furthermore, Q need not be irreducible under these conditions – it is sufficient that Q has no transient states (so that it has a steady-state probability vector ξ with positive components). Under condition (3), T and ΠT need not be contractions with respect to $\|\cdot\|_\xi$. For a counterexample, take $a_{i,i+1} = 1$ for $i = 1, \ldots, n-1$, and $a_{n,1} = 1/2$, with every other entry of A equal to 0. Take also $q_{i,i+1} = 1$ for $i = 1, \ldots, n-1$,

† In the following argument, the projection Πz of a complex vector z is obtained by separately projecting the real and the imaginary components of z on S. The projection norm for a complex vector $x + iy$ is defined by

$$\|x + iy\|_\xi = \sqrt{\|x\|_\xi^2 + \|y\|_\xi^2}.$$

and $q_{n,1} = 1$, with every other entry of Q equal to 0, so $\xi_i = 1/n$ for all i. Then for $z = (0, 1, \ldots, 1)'$ we have $Az = (1, \ldots, 1, 0)'$ and $\|Az\|_\xi = \|z\|_\xi$, so A is not a contraction with respect to $\|\cdot\|_\xi$. Taking S to be the entire space $\Re^n$, we see that the same is true for ΠA.

When the row sums of $|A|$ are no greater than one, one can construct Q with $|A| \leq Q$ by adding another matrix to $|A|$:

$$Q = |A| + Diag(e - |A|e)R, \tag{6.259}$$

where R is a transition probability matrix, e is the unit vector that has all components equal to 1, and $Diag(e - |A|e)$ is the diagonal matrix with $1 - \sum_{m=1}^n |a_{im}|$, $i = 1, \ldots, n$, on the diagonal. Then the row sum deficit of the ith row of A is distributed to the columns j according to fractions r_{ij}, the components of R.

The next proposition uses different assumptions than Prop. 6.8.2, and applies to cases where there is no special index $\overline{i}$ such that $\sum_{j=1}^n |a_{\overline{i}j}| < 1$. In fact A may itself be a transition probability matrix, so that $I - A$ need not be invertible, and the original system may have multiple solutions; see the subsequent Example 6.8.2. The proposition suggests the use of a damped version of the T mapping in various methods (compare with Section 6.7 and the average cost case for $\lambda = 0$).

Proposition 6.8.3: Assume that there are no transient states corresponding to Q, that ξ is a steady-state probability vector of Q, and that $|A| \leq Q$. Assume further that $I - \Pi A$ is invertible. Then the mapping ΠT_γ, where

$$T_\gamma = (1 - \gamma)I + \gamma T,$$

is a contraction with respect to $\|\cdot\|_\xi$ for all $\gamma \in (0, 1)$.

Proof: The argument of the proof of Prop. 6.8.2 shows that the condition $|A| \leq Q$ implies that A is nonexpansive with respect to the norm $\|\cdot\|_\xi$. Furthermore, since $I - \Pi A$ is invertible, we have $z \neq \Pi Az$ for all $z \neq 0$. Hence for all $\gamma \in (0, 1)$ and $z \in \Re^n$,

$$\|(1-\gamma)z+\gamma\Pi Az\|_\xi < (1-\gamma)\|z\|_\xi+\gamma\|\Pi Az\|_\xi \leq (1-\gamma)\|z\|_\xi+\gamma\|z\|_\xi = \|z\|_\xi, \tag{6.260}$$

where the strict inequality follows from the strict convexity of the norm, and the weak inequality follows from the nonexpansiveness of ΠA. If we define

$$\rho_\gamma = \sup\big\{\|(1 - \gamma)z + \gamma\Pi Az\|_\xi \mid \|z\| \leq 1\big\},$$

and note that the supremum above is attained by Weierstrass' Theorem, we see that Eq. (6.260) yields $\rho_\gamma < 1$ and

$$\|(1-\gamma)z + \gamma\Pi Az\|_\xi \leq \rho_\gamma \|z\|_\xi, \qquad \forall\ z \in \Re^n.$$

From the definition of T_γ, we have for all $x, y \in \Re^n$,

$$\begin{aligned}\Pi T_\gamma x - \Pi T_\gamma y &= \Pi T_\gamma (x-y) = (1-\gamma)\Pi(x-y) + \gamma\Pi A(x-y) \\ &= (1-\gamma)\Pi(x-y) + \gamma\Pi\big(\Pi A(x-y)\big),\end{aligned}$$

so defining $z = x - y$, and using the preceding two relations and the nonexpansiveness of Π, we obtain

$$\begin{aligned}\|\Pi T_\gamma x - \Pi T_\gamma y\|_\xi &= \|(1-\gamma)\Pi z + \gamma\Pi(\Pi Az)\|_\xi \leq \|(1-\gamma)z + \gamma\Pi Az\|_\xi \\ &\leq \rho_\gamma \|z\|_\xi = \rho_\gamma \|x-y\|_\xi,\end{aligned}$$

for all $x, y \in \Re^n$. **Q.E.D.**

Note that the mappings ΠT_γ and ΠT have the same fixed points, so under the assumptions of Prop. 6.8.3, there is a unique fixed point Φr^* of ΠT. We now discuss examples of choices of ξ and Q in some special cases.

Example 6.8.1 (Discounted DP Problems and Exploration)

Bellman's equation for the cost vector of a stationary policy in an n-state discounted DP problem has the form $x = T(x)$, where

$$T(x) = \alpha P x + g,$$

g is the vector of single-stage costs associated with the n states, P is the transition probability matrix of the associated Markov chain, and $\alpha \in (0,1)$ is the discount factor. If P is an irreducible Markov chain, and ξ is chosen to be its unique steady-state probability vector, the matrix inversion method based on Eq. (6.249) becomes LSTD(0). The methodology of the present section also allows row sampling/state sequence generation using a Markov chain $\overline{P}$ other than P, with an attendant change in ξ, as discussed in the context of exploration-enhanced methods in Section 6.3.7.

Example 6.8.2 (Undiscounted DP Problems)

Consider the equation $x = Ax + b$, for the case where A is a substochastic matrix ($a_{ij} \geq 0$ for all i, j and $\sum_{j=1}^n a_{ij} \leq 1$ for all i). Here $1 - \sum_{j=1}^n a_{ij}$ may be viewed as a transition probability from state i to some absorbing state denoted 0. This is Bellman's equation for the cost vector of a stationary policy of a SSP. If the policy is proper in the sense that from any state $i \neq 0$

there exists a path of positive probability transitions from i to the absorbing state 0, the matrix

$$Q = |A| + Diag(e - |A|e)R$$

[cf. Eq. (6.259)] is irreducible, provided R has positive components. As a result, the conditions of Prop. 6.8.2 under condition (2) are satisfied, and T and ΠT are contractions with respect to $\|\cdot\|_\xi$. It is also possible to use a matrix R whose components are not all positive, as long as Q is irreducible, in which case Prop. 6.8.2 under condition (3) applies (cf. Prop. 6.7.1).

Consider also the equation $x = Ax + b$ for the case where A is an irreducible transition probability matrix, with steady-state probability vector ξ. This is related to Bellman's equation for the differential cost vector of a stationary policy of an average cost DP problem involving a Markov chain with transition probability matrix A. Then, if the unit vector e is not contained in the subspace S spanned by the basis functions, the matrix $I - \Pi A$ is invertible, as shown in Section 6.7. As a result, Prop. 6.8.3 applies and shows that the mapping $(1-\gamma)I + \gamma A$, is a contraction with respect to $\|\cdot\|_\xi$ for all $\gamma \in (0,1)$ (cf. Section 6.7, Props. 6.7.1, 6.7.2).

The projected equation methodology of this section applies to general linear fixed point equations, where A need not have a probabilistic structure. A class of such equations where ΠA is a contraction is given in the following example, an important case in the field of numerical methods/scientific computation where iterative methods are used for solving linear equations.

Example 6.8.3 (Weakly Diagonally Dominant Systems)

Consider the solution of the system

$$Cx = d,$$

where $d \in \Re^n$ and C is an $n \times n$ matrix that is weakly diagonally dominant, i.e., its components satisfy

$$c_{ii} \neq 0, \qquad \sum_{j \neq i} |c_{ij}| \leq |c_{ii}|, \qquad i = 1, \ldots, n. \tag{6.261}$$

By dividing the ith row by c_{ii}, we obtain the equivalent system $x = Ax + b$, where the components of A and b are

$$a_{ij} = \begin{cases} 0 & \text{if } i = j, \\ -\frac{c_{ij}}{c_{ii}} & \text{if } i \neq j, \end{cases} \qquad b_i = \frac{d_i}{c_{ii}}, \qquad i = 1, \ldots, n.$$

Then, from Eq. (6.261), we have

$$\sum_{j=1}^{n} |a_{ij}| = \sum_{j \neq i} \frac{|c_{ij}|}{|c_{ii}|} \leq 1, \qquad i = 1, \ldots, n,$$

so Props. 6.8.2 and 6.8.3 may be used under the appropriate conditions. In particular, if the matrix Q given by Eq. (6.259) has no transient states and there exists an index $\overline{i}$ such that $\sum_{j=1}^{n} |a_{\overline{i}j}| < 1$, Prop. 6.8.2 applies and shows that ΠT is a contraction.

Alternatively, instead of Eq. (6.261), assume the somewhat more restrictive condition

$$|1 - c_{ii}| + \sum_{j \neq i} |c_{ij}| \leq 1, \qquad i = 1, \ldots, n, \tag{6.262}$$

and consider the equivalent system $x = Ax + b$, where

$$A = I - C, \qquad b = d.$$

Then, from Eq. (6.262), we have

$$\sum_{j=1}^{n} |a_{ij}| = |1 - c_{ii}| + \sum_{j \neq i} |c_{ij}| \leq 1, \qquad i = 1, \ldots, n,$$

so again Props. 6.8.2 and 6.8.3 apply under appropriate conditions.

Let us finally address the question whether it is possible to find Q such that $|A| \leq Q$ and the corresponding Markov chain has no transient states or is irreducible. To this end, assume that $\sum_{j=1}^{n} |a_{ij}| \leq 1$ for all i. If A is itself irreducible, then any Q such that $|A| \leq Q$ is also irreducible. Otherwise, consider the set

$$\overline{I} = \left\{ i \;\middle|\; \sum_{j=1}^{n} |a_{ij}| < 1 \right\},$$

and assume that it is nonempty (otherwise the only possibility is $Q = |A|$). Let $\tilde{I}$ be the set of i such that there exists a sequence of nonzero components $a_{ij_1}, a_{j_1 j_2}, \ldots, a_{j_m \overline{i}}$ such that $\overline{i} \in \overline{I}$, and let $\hat{I} = \{i \mid i \notin \overline{I} \cup \tilde{I}\}$ (we allow here the possibility that $\tilde{I}$ or $\hat{I}$ may be empty). Note that the square submatrix of $|A|$ corresponding to $\hat{I}$ is a transition probability matrix, and that we have $a_{ij} = 0$ for all $i \in \hat{I}$ and $j \notin \hat{I}$. Then it can be shown that there exists Q with $|A| \leq Q$ and no transient states if and only if the Markov chain corresponding to $\hat{I}$ has no transient states. Furthermore, there exists an irreducible Q with $|A| \leq Q$ if and only if $\hat{I}$ is empty.

6.8.4 Multistep Methods

We now consider the possibility of replacing T with a multistep mapping that has the same fixed point, such as T^ℓ with $\ell > 1$, or $T^{(\lambda)}$ given by

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

where $\lambda \in (0,1)$. For example, the LSTD(λ), LSPE(λ), and TD(λ) methods for approximate policy evaluation are based on this possibility. The key idea in extending these methods to general linear systems is that the ith component $(A^m b)(i)$ of a vector of the form $A^m b$, where $b \in \Re^n$, can be computed by averaging over properly weighted simulation-based sample values.

In multistep methods, it turns out that for technical efficiecy reasons it is important to use the same probabilistic mechanism for row and for column sampling. In particular, we generate the index sequence $\{i_0, i_1, \ldots\}$ and the transition sequence $\big\{(i_0,i_1),(i_1,i_2),\ldots\big\}$ by *using the same irreducible transition matrix* P, so ξ is the steady-state probability distribution of P. We then form the average of $w_{k,m} b_{i_{k+m}}$ over all indices k such that $i_k = i$, where

$$w_{k,m} = \begin{cases} \frac{a_{i_k i_{k+1}}}{p_{i_k i_{k+1}}} \frac{a_{i_{k+1} i_{k+2}}}{p_{i_{k+1} i_{k+2}}} \cdots \frac{a_{i_{k+m-1} i_{k+m}}}{p_{i_{k+m-1} i_{k+m}}} & \text{if } m \geq 1, \\ 1 & \text{if } m = 0. \end{cases} \tag{6.263}$$

We claim that the following is a valid approximation of $(A^m b)(i)$:

$$(A^m b)(i) \approx \frac{\sum_{k=0}^{t} \delta(i_k = i) w_{k,m} b_{i_{k+m}}}{\sum_{k=0}^{t} \delta(i_k = i)}. \tag{6.264}$$

The justification is that by the irreducibility of the associated Markov chain, we have

$$\lim_{t\to\infty} \frac{\sum_{k=0}^{t} \delta(i_k = i, i_{k+1} = j_1, \ldots, i_{k+m} = j_m)}{\sum_{k=0}^{t} \delta(i_k = i)} = p_{i j_1} p_{j_1 j_2} \cdots p_{j_{m-1} j_m}, \tag{6.265}$$

and the limit of the right-hand side of Eq. (6.264) can be written as

$$\begin{aligned} &\lim_{t\to\infty} \frac{\sum_{k=0}^{t} \delta(i_k = i) w_{k,m} b_{i_{k+m}}}{\sum_{k=0}^{t} \delta(i_k = i)} \\ &= \lim_{t\to\infty} \frac{\sum_{k=0}^{t} \sum_{j_1=1}^{n} \cdots \sum_{j_m=1}^{n} \delta(i_k = i, i_{k+1} = j_1, \ldots, i_{k+m} = j_m) w_{k,m} b_{i_{k+m}}}{\sum_{k=0}^{t} \delta(i_k = i)} \\ &= \sum_{j_1=1}^{n} \cdots \sum_{j_m=1}^{n} \lim_{t\to\infty} \frac{\sum_{k=0}^{t} \delta(i_k = i, i_{k+1} = j_1, \ldots, i_{k+m} = j_m)}{\sum_{k=0}^{t} \delta(i_k = i)} w_{k,m} b_{i_{k+m}} \\ &= \sum_{j_1=1}^{n} \cdots \sum_{j_m=1}^{n} a_{i j_1} a_{j_1 j_2} \cdots a_{j_{m-1} j_m} b_{j_m} \\ &= (A^m b)(i), \end{aligned}$$

where the third equality follows using Eqs. (6.263) and (6.265).

By using the approximation formula (6.264), it is possible to construct complex simulation-based approximations to formulas that involve powers

of A. As an example that we have not encountered so far in the DP context, we may obtain by simulation the solution x^* of the linear system $x = b+Ax$, which can be expressed as

$$x^* = (I - A)^{-1}b = \sum_{\ell=0}^{\infty} A^\ell b,$$

assuming the eigenvalues of A are all within the unit circle. Historically, this is the first method for simulation-based matrix inversion and solution of linear systems, due to von Neumann and Ulam (unpublished but described by Forsythe and Leibler [FoL50]).

λ-Methods

We will now summarize extensions of LSTD(λ), LSPE(λ), and TD(λ) to solve the general fixed point problem $x = b + Ax$. The underlying idea is the approximation of Eq. (6.264). We refer to [BeY09], [Ber11a], [Yu10a], and [Yu10b] for detailed derivations and analysis. Similar to Section 6.3.6, these methods aim to solve the λ-projected equation

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

or equivalently

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

where

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

with

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

by using simulation-based approximations, and either matrix inversion or iteration.

As in Sections 7.3.1, the simulation is used to construct approximations $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ of $C^{(\lambda)}$ and $d^{(\lambda)}$, respectively. Given the simulated sequence $\{i_0, i_1, \ldots\}$ obtained by row/column sampling using transition probabilities p_{ij}, $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ are generated by

$$C_k^{(\lambda)} = (1-\delta_k) C_{k-1}^{(\lambda)} + \delta_k z_k \left(\phi(i_k) - \frac{a_{i_k i_{k+1}}}{p_{i_k i_{k+1}}}\phi(i_{k+1})\right)',$$

$$d_k^{(\lambda)} = (1-\delta_k) d_{k-1}^{(\lambda)} + \delta_k z_k g(i_k, i_{k+1}),$$

where z_k are modified eligibility vectors given by

$$z_k = \lambda \frac{a_{i_{k-1} i_k}}{p_{i_{k-1} i_k}} z_{k-1} + \phi(i_k), \tag{6.266}$$

the initial conditions are $z_{-1} = 0$, $C_{-1}^{(\lambda)} = 0$, $d_{-1}^{(\lambda)} = 0$, and

$$\delta_k = \frac{1}{k+1}, \quad k = 0, 1, \ldots.$$

The matrix inversion/LSTD(λ) analog is to solve the equation $C_k^{(\lambda)} r = d_k^{(\lambda)}$, while the iterative/LSPE(λ) analog is

$$r_{k+1} = r_k - \gamma G_k (C_k^{(\lambda)} r_k - d_k^{(\lambda)}),$$

where G_k is a positive definite scaling matrix and γ is a positive stepsize. There is also a generalized version of the TD(λ) method. It has the form

$$r_{k+1} = r_k + \gamma_k z_k q_k(i_k),$$

where γ_k is a diminishing positive scalar stepsize, z_k is given by Eq. (6.266), and $q_k(i_k)$ is the temporal difference analog given by

$$q_k(i_k) = b_{i_k} + \frac{a_{i_{k-1} i_k}}{p_{i_{k-1} i_k}} \phi(i_{k+1})' r_k - \phi(i_k)' r_k.$$

6.8.5 Extension of Q-Learning for Optimal Stopping

If the mapping T is nonlinear (as for example in the case of multiple policies) the projected equation $\Phi r = \Pi T(\Phi r)$ is also nonlinear, and may have one or multiple solutions, or no solution at all. On the other hand, if ΠT is a contraction, there is a unique solution. We have seen in Section 6.5.3 a nonlinear special case of projected equation where ΠT is a contraction, namely optimal stopping. This case can be generalized as we now show.

Let us consider a system of the form

$$x = T(x) = Af(x) + b, \tag{6.267}$$

where $f : \Re^n \mapsto \Re^n$ is a mapping with scalar function components of the form $f(x) = \big(f_1(x_1), \ldots, f_n(x_n)\big)$. We assume that each of the mappings $f_i : \Re \mapsto \Re$ is nonexpansive in the sense that

$$\big|f_i(x_i) - f_i(\bar{x}_i)\big| \leq |x_i - \bar{x}_i|, \qquad \forall\ i = 1, \ldots, n,\ x_i, \bar{x}_i \in \Re. \tag{6.268}$$

This guarantees that T is a contraction mapping with respect to any norm $\|\cdot\|$ with the property

$$\|y\| \leq \|z\| \qquad \text{if } |y_i| \leq |z_i|, \qquad \forall\ i = 1, \ldots, n,$$

whenever A is a contraction with respect to that norm. Such norms include weighted l_1 and l_∞ norms, the norm $\|\cdot\|_\xi$, as well as any scaled Euclidean

norm $\|x\| = \sqrt{x'Dx}$, where D is a positive definite symmetric matrix with nonnegative components. Under the assumption (6.268), the theory of Section 6.8.2 applies and suggests appropriate choices of a Markov chain for simulation so that ΠT is a contraction.

As an example, consider the equation

$$x = T(x) = \alpha P f(x) + b,$$

where P is an irreducible transition probability matrix with steady-state probability vector ξ, $\alpha \in (0,1)$ is a scalar discount factor, and f is a mapping with components

$$f_i(x_i) = \min\{c_i, x_i\}, \qquad i = 1, \ldots, n, \tag{6.269}$$

where c_i are some scalars. This is the Q-factor equation corresponding to a discounted optimal stopping problem with states $i = 1, \ldots, n$, and a choice between two actions at each state i: stop at a cost c_i, or continue at a cost b_i and move to state j with probability p_{ij}. The optimal cost starting from state i is $\min\{c_i, x_i^*\}$, where x^* is the fixed point of T. As a special case of Prop. 6.8.2, we obtain that ΠT is a contraction with respect to $\|\cdot\|_\xi$. Similar results hold in the case where αP is replaced by a matrix A satisfying condition (2) of Prop. 6.8.2, or the conditions of Prop. 6.8.3.

A version of the LSPE-type algorithm for solving the system (6.267), which extends the method of Section 6.5.3 for optimal stopping, may be used when ΠT is a contraction. In particular, the iteration

$$\Phi r_{k+1} = \Pi T(\Phi r_k), \qquad k = 0, 1, \ldots,$$

takes the form

$$r_{k+1} = \left(\sum_{i=1}^{n} \xi_i\, \phi(i)\phi(i)'\right)^{-1} \sum_{i=1}^{n} \xi_i\, \phi(i) \left(\sum_{j=1}^{n} a_{ij} f_j\big(\phi(j)'r_k\big) + b_i\right),$$

and is approximated by

$$r_{k+1} = \left(\sum_{t=0}^{k} \phi(i_t)\phi(i_t)'\right)^{-1} \sum_{t=0}^{k} \phi(i_t) \left(\frac{a_{i_t j_t}}{p_{i_t j_t}} f_{j_t}\big(\phi(j_t)'r_k\big) + b_{i_t}\right). \tag{6.270}$$

Here, as before, $\{i_0, i_1, \ldots\}$ is a state sequence, and $\{(i_0, j_0), (i_1, j_1), \ldots\}$ is a transition sequence satisfying Eqs. (6.246) and (6.248) with probability 1. The justification of this approximation is very similar to the ones given so far, and will not be discussed further. Diagonally scaled versions of this iteration are also possible.

A difficulty with iteration (6.270) is that the terms $f_{j_t}\big(\phi(j_t)'r_k\big)$ must be computed for all $t = 0, \ldots, k$, at every step k, thereby resulting in

significant overhead. The methods to bypass this difficulty in the case of optimal stopping, discussed at the end of Section 6.5.3, can be extended to the more general context considered here.

Let us finally consider the case where instead of $A = \alpha P$, the matrix A satisfies condition (2) of Prop. 6.8.2, or the conditions of Prop. 6.8.3. The case where $\sum_{j=1}^{n} |a_{\overline{i}j}| < 1$ for some index $\overline{i}$, and $0 \leq A \leq Q$, where Q is an irreducible transition probability matrix, corresponds to an undiscounted optimal stopping problem where the stopping state will be reached from all other states with probability 1, even without applying the stopping action. In this case, from Prop. 6.8.2 under condition (3), it follows that ΠA is a contraction with respect to some norm, and hence $I - \Pi A$ is invertible. Using this fact, it can be shown by modifying the proof of Prop. 6.8.3 that the mapping ΠT_γ, where

$$T_\gamma(x) = (1-\gamma)x + \gamma T(x)$$

is a contraction with respect to $\|\cdot\|_\xi$ for all $\gamma \in (0,1)$. Thus, ΠT_γ has a unique fixed point, and must be also the unique fixed point of ΠT (since ΠT and ΠT_γ have the same fixed points).

In view of the contraction property of T_γ, the "damped" PVI iteration

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

converges to the unique fixed point of ΠT and takes the form

$$r_{k+1} = (1-\gamma)r_k + \gamma \left(\sum_{i=1}^{n} \xi_i \, \phi(i)\phi(i)' \right)^{-1} \sum_{i=1}^{n} \xi_i \, \phi(i) \left(\sum_{j=1}^{n} a_{ij} f_j\big(\phi(j)' r_k\big) + b_i \right)$$

As earlier, it can be approximated by the LSPE iteration

$$r_{k+1} = (1-\gamma)r_k + \gamma \left(\sum_{t=0}^{k} \phi(i_t)\phi(i_t)' \right)^{-1} \sum_{t=0}^{k} \phi(i_t) \left(\frac{a_{i_t j_t}}{p_{i_t j_t}} f_{j_t}\big(\phi(j_t)' r_k\big) + b_{i_t} \right)$$

[cf. Eq. (6.270)].

6.8.6 Bellman Equation Error-Type Methods

We will now consider an alternative approach for approximate solution of the linear equation $x = T(x) = b + Ax$, based on finding a vector r that minimizes

$$\|\Phi r - T(\Phi r)\|_\xi^2,$$

or

$$\sum_{i=1}^{n} \xi_i \left(\phi(i)' r - \sum_{j=1}^{n} a_{ij} \phi(j)' r - b_i \right)^2,$$

where ξ is a distribution with positive components. In the DP context where the equation $x = T(x)$ is the Bellman equation for a fixed policy, this is known as the *Bellman equation error approach* (see [BeT96], Section 6.10 for a detailed discussion of this case, and the more complicated nonlinear case where T involves minimization over multiple policies). We assume that the matrix $(I - A)\Phi$ has rank s, which guarantees that the vector r^* that minimizes the weighted sum of squared errors is unique.

We note that the equation error approach is related to the projected equation approach. To see this, consider the case where ξ is the uniform distribution, so the problem is to minimize

$$\big\|\Phi r - (b + A\Phi r)\big\|^2, \tag{6.271}$$

where $\|\cdot\|$ is the standard Euclidean norm. By setting the gradient to 0, we see that a necessary and sufficient condition for optimality is

$$\Phi'(I - A)'\big(\Phi r^* - T(\Phi r^*)\big) = 0,$$

or equivalently,

$$\Phi'\big(\Phi r^* - \hat{T}(\Phi r^*)\big) = 0,$$

where

$$\hat{T}(x) = T(x) + A'\big(x - T(x)\big).$$

Thus minimization of the equation error (6.271) is equivalent to solving the projected equation

$$\Phi r = \Pi\hat{T}(\Phi r),$$

where Π denotes projection with respect to the standard Euclidean norm. A similar conversion is possible when ξ is a general distribution with positive components.

Error bounds analogous to the projected equation bounds of Eqs. (6.242) and (6.243) can be developed for the equation error approach, assuming that $I - A$ is invertible and x^* is the unique solution. In particular, let $\tilde{r}$ minimize $\|\Phi r - T(\Phi r)\|_\xi^2$. Then

$$x^* - \Phi\tilde{r} = Tx^* - T(\Phi\tilde{r}) + T(\Phi\tilde{r}) - \Phi\tilde{r} = A(x^* - \Phi\tilde{r}) + T(\Phi\tilde{r}) - \Phi\tilde{r},$$

so that

$$x^* - \Phi\tilde{r} = (I - A)^{-1}\big(T(\Phi\tilde{r}) - \Phi\tilde{r}\big).$$

Thus, we obtain

$$\begin{aligned}
\|x^* - \Phi\tilde{r}\|_\xi &\le \big\|(I - A)^{-1}\big\|_\xi \|\Phi\tilde{r} - T(\Phi\tilde{r})\|_\xi \\
&\le \big\|(I - A)^{-1}\big\|_\xi \big\|\Pi x^* - T(\Pi x^*)\big\|_\xi \\
&= \big\|(I - A)^{-1}\big\|_\xi \big\|\Pi x^* - x^* + Tx^* - T(\Pi x^*)\big\|_\xi \\
&= \big\|(I - A)^{-1}\big\|_\xi \big\|(I - A)(\Pi x^* - x^*)\big\|_\xi \\
&\le \big\|(I - A)^{-1}\big\|_\xi \|I - A\|_\xi \|x^* - \Pi x^*\|_\xi,
\end{aligned}$$

where the second inequality holds because $\tilde{r}$ minimizes $\|\Phi r - T(\Phi r)\|_\xi^2$. In the case where T is a contraction mapping with respect to the norm $\|\cdot\|_\xi$, with modulus $\alpha \in (0,1)$, a similar calculation yields

$$\|x^* - \Phi\tilde{r}\|_\xi \leq \frac{1+\alpha}{1-\alpha}\,\|x^* - \Pi x^*\|_\xi.$$

The vector r^* that minimizes $\|\Phi r - T(\Phi r)\|_\xi^2$ satisfies the corresponding necessary optimality condition

$$\begin{aligned}\sum_{i=1}^n \xi_i \left(\phi(i) - \sum_{j=1}^n a_{ij}\phi(j)\right)\left(\phi(i) - \sum_{j=1}^n a_{ij}\phi(j)\right)' r^* \\ = \sum_{i=1}^n \xi_i \left(\phi(i) - \sum_{j=1}^n a_{ij}\phi(j)\right) b_i.\end{aligned} \tag{6.272}$$

To obtain a simulation-based approximation to Eq. (6.272), without requiring the calculation of row sums of the form $\sum_{j=1}^n a_{ij}\phi(j)$, we introduce an additional sequence of transitions $\{(i_0, j_0'), (i_1, j_1'), \ldots\}$ (see Fig. 6.8.2), which is generated according to the transition probabilities p_{ij} of the Markov chain, and is "independent" of the sequence $\{(i_0, j_0), (i_1, j_1), \ldots\}$ in the sense that with probability 1,

$$\lim_{t\to\infty} \frac{\sum_{k=0}^t \delta(i_k = i, j_k = j)}{\sum_{k=0}^t \delta(i_k = i)} = \lim_{t\to\infty} \frac{\sum_{k=0}^t \delta(i_k = i, j_k' = j)}{\sum_{k=0}^t \delta(i_k = i)} = p_{ij}, \tag{6.273}$$

for all $i, j = 1, \ldots, n$, and

$$\lim_{t\to\infty} \frac{\sum_{k=0}^t \delta(i_k = i, j_k = j, j_k' = j')}{\sum_{k=0}^t \delta(i_k = i)} = p_{ij}p_{ij'}, \tag{6.274}$$

for all $i, j, j' = 1, \ldots, n$. At time t, we form the linear equation

$$\begin{aligned}\sum_{k=0}^t \left(\phi(i_k) - \frac{a_{i_k j_k}}{p_{i_k j_k}}\phi(j_k)\right)\left(\phi(i_k) - \frac{a_{i_k j_k'}}{p_{i_k j_k'}}\phi(j_k')\right)' r \\ = \sum_{k=0}^t \left(\phi(i_k) - \frac{a_{i_k j_k}}{p_{i_k j_k}}\phi(j_k)\right) b_{i_k}.\end{aligned} \tag{6.275}$$

Similar to our earlier analysis, it can be seen that this is a valid approximation to Eq. (6.272).

Note a disadvantage of this approach relative to the projected equation approach (cf. Section 6.8.1). It is necessary to generate two sequences of transitions (rather than one). Moreover, both of these sequences enter

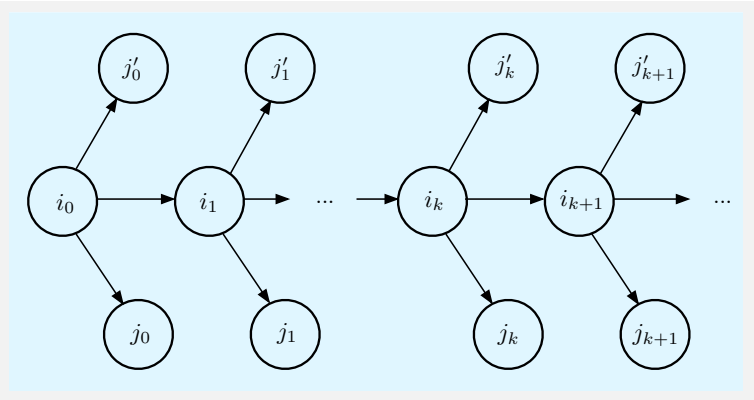

Figure 6.8.2 A possible simulation mechanism for minimizing the equation error norm [cf. Eq. (6.275)]. We generate a sequence of states $\{i_0, i_1, \ldots\}$ according to the distribution ξ, by simulating a single infinitely long sample trajectory of the chain. Simultaneously, we generate two independent sequences of transitions, $\{(i_0, j_0), (i_1, j_1), \ldots\}$ and $\{(i_0, j'_0), (i_1, j'_1), \ldots\}$, according to the transition probabilities p_{ij}, so that Eqs. (6.273) and (6.274) are satisfied.

Eq. (6.275), which thus contains more simulation noise than its projected equation counterpart [cf. Eq. (6.249)].

Let us finally note that the equation error approach can be generalized to yield a simulation-based method for solving the general linear least squares problem

$$\min_r \sum_{i=1}^n \xi_i \left(c_i - \sum_{j=1}^m q_{ij} \phi(j)' r \right)^2,$$

where q_{ij} are the components of an $n \times m$ matrix Q, and c_i are the components of a vector $c \in \Re^n$. In particular, one may write the corresponding optimality condition [cf. Eq. (6.272)] and then approximate it by simulation [cf. Eq. (6.275)]; see [BeY09], and [WPB09], [PWB09], which also discuss a regression-based approach to deal with nearly singular problems (cf. the regression-based LSTD method of Section 6.3.4). Conversely, one may consider a selected set I of states of moderate size, and find r^* that minimizes the sum of squared Bellman equation errors only for these states:

$$r^* \in \arg\min_{r \in \Re^s} \sum_{i \in I} \xi_i \left(\phi(i)' r - \sum_{j=1}^n a_{ij} \phi(j)' r - b_i \right)^2.$$

This least squares problem may be solved by conventional (non-simulation) methods.

An interesting question is how the approach of this section compares with the projected equation approach in terms of approximation error. No definitive answer seems possible, and examples where one approach gives better results than the other have been constructed. Reference [Ber95] shows that in the example of Exercise 6.9, the projected equation approach gives worse results. For an example where the projected equation approach may be preferable, see Exercise 6.11.

Approximate Policy Iteration with Bellman Equation Error Evaluation

When the Bellman equation error approach is used in conjunction with approximate policy iteration in a DP context, it is susceptible to chattering and oscillation just as much as the projected equation approach (cf. Section 6.3.8). The reason is that both approaches operate within the same greedy partition, and oscillate when there is a cycle of policies $\mu^k, \mu^{k+1}, \ldots, \mu^{k+m}$ with

$$r_{\mu^k} \in R_{\mu^{k+1}},\ r_{\mu^{k+1}} \in R_{\mu^{k+2}}, \ldots, r_{\mu^{k+m-1}} \in R_{\mu^{k+m}},\ r_{\mu^{k+m}} \in R_{\mu^k}$$

(cf. Fig. 6.3.4). The only difference is that the weight vector r_μ of a policy μ is calculated differently (by solving a least-squares Bellman error problem versus solving a projected equation). In practice the weights calculated by the two approaches may differ somewhat, but generally not enough to cause dramatic changes in qualitative behavior. Thus, much of our discussion of optimistic policy iteration in Sections 6.3.5-6.3.6 applies to the Bellman equation error approach as well.

Example 6.3.2 (continued)

Let us return to Example 6.3.2 where chattering occurs when r_μ is evaluated using the projected equation. When the Bellman equation error approach is used instead, the greedy partition remains the same (cf. Fig. 6.3.6), the weight of policy μ is $r_\mu = 0$ (as in the projected equation case), and for $p \approx 1$, the weight of policy μ^* can be calculated to be

$$r_{\mu^*} \approx \frac{c}{(1-\alpha)\big((1-\alpha)^2 + (2-\alpha)^2\big)}$$

[which is almost the same as the weight $c/(1-\alpha)$ obtained in the projected equation case]. Thus with both approaches we have oscillation between μ and μ^* in approximate policy iteration, and chattering in optimistic versions, with very similar iterates.

6.8.7 Oblique Projections

Some of the preceding methodology regarding projected equations can be generalized to the case where the projection operator Π is oblique (i.e., it is not a projection with respect to the weighted Euclidean norm, see e.g., Saad [Saa03]). Such projections have the form

$$\Pi = \Phi(\Psi'\Xi\Phi)^{-1}\Psi'\Xi, \tag{6.276}$$

where as before, Ξ is the diagonal matrix with the components $\xi_1, \ldots, \xi_n$ of a positive distribution vector ξ along the diagonal, Φ is an $n \times s$ matrix of rank s, and Ψ is an $n \times s$ matrix of rank s. The earlier case corresponds to $\Psi = \Phi$. Two characteristic properties of Π as given by Eq. (6.276) are that its range is the subspace $S = \{\Phi r \mid r \in \Re^s\}$ and that it is idempotent, i.e., $\Pi^2 = \Pi$. Conversely, a matrix Π with these two properties can be shown to have the form (6.276) for some $n \times s$ matrix Ψ of rank s and a diagonal matrix Ξ with the components $\xi_1, \ldots, \xi_n$ of a positive distribution vector ξ along the diagonal. Oblique projections arise in a variety of interesting contexts, for which we refer to the literature.

Let us now consider the generalized projected equation

$$\Phi r = \Pi T(\Phi r) = \Pi(b + A\Phi r). \tag{6.277}$$

Using Eq. (6.276) and the fact that Φ has rank s, it can be written as

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

or equivalently $\Psi'\Xi\Phi r = \Psi'\Xi(b + A\Phi r)$, which can be finally written as

$$Cr = d,$$

where

$$C = \Psi'\Xi(I - A)\Phi, \qquad d = \Psi'\Xi b. \tag{6.278}$$

These equations should be compared to the corresponding equations for the Euclidean projection case where $\Psi = \Phi$ [cf. Eq. (6.244)].

It is clear that row and column sampling can be adapted to provide simulation-based estimates C_k and d_k of C and d, respectively. The corresponding equations have the form [cf. Eq. (6.249)]

$$C_k = \frac{1}{k+1}\sum_{t=0}^{k}\psi(i_t)\left(\phi(i_t) - \frac{a_{i_t j_t}}{p_{i_t j_t}}\phi(j_t)\right)', \quad d_k = \frac{1}{k+1}\sum_{t=0}^{k}\psi(i_t)b_{i_t}, \tag{6.279}$$

where $\psi'(i)$ is the ith row of Ψ. The sequence of vectors $C_k^{-1}d_k$ converges with probability one to the solution $C^{-1}d$ of the projected equation, assuming that C is nonsingular. For cases where C_k is nearly singular, the

regression/regularization-based estimate (6.251) may be used. The corresponding iterative method is

$$r_{k+1} = (C_k' \Sigma_k^{-1} C_k + \beta I)^{-1} (C_k' \Sigma_k^{-1} d_k + \beta r_k),$$

and can be shown to converge with probability one to $C^{-1}d$.

An example where oblique projections arise in DP is aggregation/discretization with a coarse grid [cases (c) and (d) in Section 6.4, with the aggregate states corresponding some distinct representative states $\{x_1, \ldots, x_s\}$ of the original problem; also Example 6.4.1]. Then the aggregation equation for a discounted problem has the form

$$\Phi r = \Phi D(b + \alpha P \Phi r), \tag{6.280}$$

where the rows of D are unit vectors (have a single component equal to 1, corresponding to a representative state, and all other components equal to 0), and the rows of Φ are probability distributions, with the rows corresponding to the representative states x_k having a single unit component, $\Phi_{x_k x_k} = 1$, $k = 1, \ldots, s$. Then the matrix $D\Phi$ can be seen to be the identity, so we have $\Phi D \cdot \Phi D = \Phi D$ and it follows that ΦD is an oblique projection. The conclusion is that the aggregation equation (6.280) in the special case of coarse grid discretization is the projected equation (6.277), with the oblique projection $\Pi = \Phi D$.

6.8.8 Generalized Aggregation by Simulation

We will finally discuss the simulation-based iterative solution of a general system of equations of the form

$$r = DT(\Phi r), \tag{6.281}$$

where $T : \Re^n \mapsto \Re^m$ is a (possibly nonlinear) mapping, D is an $s \times m$ matrix, and Φ is an $n \times s$ matrix. In the case $m = n$, we can regard the system (6.281) as an approximation to a system of the form

$$x = T(x). \tag{6.282}$$

In particular, the variables x_i of the system (6.282) are approximated by linear combinations of the variables r_j of the system (6.281), using the rows of Φ. Furthermore, the components of the mapping DT are obtained by linear combinations of the components of T, using the rows of D. Thus we may view the system (6.281) as being obtained by aggregation/linear combination of the variables and the equations of the system (6.282).

We have encountered equations of the form (6.281) in our discussion of aggregation (Section 6.4) and Q-learning (Section 6.5). For example, the

aggregation mapping

$$(FR)(x) = \sum_{i=1}^{n} d_{xi} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \sum_{y \in S} \phi_{jy} R(y) \right), \quad x \in S, \tag{6.283}$$

[cf. Eq. (6.162)] is of the form (6.281), where $r = R$, the dimension s is equal to the number of aggregate states x, $m = n$ is the number of states i, and the matrices D and Φ consist of the disaggregation and aggregation probabilities, respectively.

As another example the Q-learning mapping

$$(FQ)(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \min_{v \in U(j)} Q(j,v) \right), \qquad \forall\ (i,u), \tag{6.284}$$

[cf. Eq. (6.180)] is of the form (6.281), where $r = Q$, the dimensions s and n are equal to the number of state-control pairs (i,u), the dimension m is the number of state-control-next state triples (i,u,j), the components of D are the appropriate probabilities $p_{ij}(u)$, Φ is the identity, and T is the nonlinear mapping that transforms Q to the vector with a component $g(i,u,j) + \alpha \min_{v \in U(j)} Q(j,v)$ for each (i,u,j).

As a third example, consider the following Bellman's equation over the space of post-decision states m [cf. Eq. (6.11)]:

$$V(m) = \sum_{j=1}^{n} q(m,j) \min_{u \in U(j)} \Big[g(j,u) + \alpha V\big(f(j,u)\big) \Big], \qquad \forall\ m. \tag{6.285}$$

This equation is of the form (6.281), where $r = V$, the dimension s is equal to the number of post-decision states x, $m = n$ is the number of (pre-decision) states i, the matrix D consists of the probabilities $q(m,j)$, and Φ is the identity matrix.

There are also versions of the preceding examples, which involve evaluation of a single policy, in which case there is no minimization in Eqs. (6.284)-(6.285), and the corresponding mapping T is linear. We will now consider separately cases where T is linear and where T is nonlinear. For the linear case, we will give an LSTD-type method, while for the nonlinear case (where the LSTD approach does not apply), we will discuss iterative methods under some contraction assumptions on T, D, and Φ.

The Linear Case

Let T be linear, so the equation $r = DT(\Phi r)$ has the form

$$r = D(b + A\Phi r), \tag{6.286}$$

where A is an $m \times n$ matrix, and $b \in \Re^s$. We can thus write this equation as

$$Er = f,$$

where

$$E = I - DA\Phi, \qquad f = Db.$$

To interpret the system (6.286), note that the matrix $A\Phi$ is obtained by replacing the n columns of A by s weighted sums of columns of A, with the weights defined by the corresponding columns of Φ. The matrix $DA\Phi$ is obtained by replacing the m rows of $A\Phi$ by s weighted sums of rows of $A\Phi$, with the weights defined by the corresponding rows of D. The simplest case is to form $DA\Phi$ by discarding $n - s$ columns and $m - s$ rows of A.

As in the case of projected equations (cf. Section 6.8.1), we can use low-dimensional simulation to approximate E and f based on row and column sampling. One way to do this is to introduce for each index $i = 1, \ldots, n$, a distribution $\{p_{ij} \mid j = 1, \ldots, m\}$ with the property

$$p_{ij} > 0 \qquad \text{if} \qquad a_{ij} \neq 0,$$

and to obtain a sample sequence $\big\{(i_0, j_0), (i_1, j_1), \ldots\big\}$. We do so by first generating a sequence of row indices $\{i_0, i_1, \ldots\}$ through sampling according to some distribution $\{\xi_i \mid i = 1, \ldots, m\}$, and then by generating for each t the column index j_t by sampling according to the distribution $\{p_{i_t j} \mid j = 1, \ldots, n\}$. There are also alternative schemes, in which we first sample rows of D and then generate rows of A, along the lines discussed in Section 6.4.2 (see also Exercise 6.14).

Given the first $k+1$ samples, we form the matrix $\hat{E}_k$ and vector $\hat{f}_k$ given by

$$\hat{E}_k = I - \frac{1}{k+1} \sum_{t=0}^{k} \frac{a_{i_t j_t}}{\xi_{i_t} p_{i_t j_t}} d(i_t)\phi(j_t)', \qquad \hat{f}_k = \frac{1}{k+1} \sum_{t=0}^{k} \frac{1}{\xi_{i_t}} d(i_t) b_t,$$

where $d(i)$ is the ith column of D and $\phi(j)'$ is the jth row of Φ. By using the expressions

$$E = I - \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} d(i)\phi(j)', \qquad f = \sum_{i=1}^{m} d(i) b_i,$$

and law of large numbers arguments, it can be shown that $\hat{E}_k \to E$ and $\hat{f}_k \to f$, similar to the case of projected equations. In particular, we can write

$$f_k = \sum_{i=1}^{m} \frac{\sum_{t=0}^{k} \delta(i_t = i)}{k+1} \frac{1}{\xi_i} d(i) b_i,$$

and since

$$\sum_{t=0}^{k} \frac{\delta(i_t = i)}{k+1} \to \xi_i,$$

we have

$$f_k \to \sum_{i=1}^{m} d(i) b_i = Db.$$

Similarly, we can write

$$\frac{1}{k+1} \sum_{t=0}^{k} \frac{a_{i_t j_t}}{p_{i_t j_t}} d(i_t)\phi(j_t)' = m \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{\sum_{t=0}^{k} \delta(i_t = i, j_t = j)}{k+1} \frac{a_{ij}}{\xi_i p_{ij}} d(i)\phi(j)',$$

and since

$$\frac{\sum_{t=0}^{k} \delta(i_t = i, j_t = j)}{k+1} \to \xi_i p_{ij},$$

we have

$$E_k \to \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} d(i)\phi(j)' = E.$$

The convergence $\hat{E}_k \to E$ and $\hat{f}_k \to f$ implies in turn that $\hat{E}_k^{-1} \hat{f}_k$ converges to the solution of the equation $r = D(b + A\Phi r)$. There is also a regression-based version of this method that is suitable for the case where $\hat{E}_k$ is nearly singular (cf. Section 6.3.4), and an iterative LSPE-type method that works even when $\hat{E}_k$ is singular [cf. Eq. (6.76)].

The Nonlinear Case

Consider now the case where T is nonlinear and has the contraction property

$$\|T(x) - T(\overline{x})\|_\infty \leq \alpha \|x - \overline{x}\|_\infty, \qquad \forall\ x \in \Re^m,$$

where α is a scalar with $0 < \alpha < 1$ and $\|\cdot\|_\infty$ denotes the sup-norm. Furthermore, let the components of the matrices D and Φ satisfy

$$\sum_{i=1}^{m} |d_{\ell i}| \leq 1, \qquad \forall\ \ell = 1, \ldots, s,$$

and

$$\sum_{\ell=1}^{s} |\phi_{j\ell}| \leq 1, \qquad \forall\ j = 1, \ldots, n.$$

These assumptions imply that D and Φ are nonexpansive in the sense that

$$\|Dx\|_\infty \leq \|x\|_\infty, \qquad \forall\ x \in \Re^n,$$

$$\|\Phi y\|_\infty \leq \|y\|_\infty, \qquad \forall\ y \in \Re^s,$$

so that $DT\Phi$ is a sup-norm contraction with modulus α, and the equation $r = DT(\Phi r)$ has a unique solution, denoted r^*.

The ideas underlying the Q-learning algorithm and its analysis (cf. Section 6.5.1) can be extended to provide a simulation-based algorithm for solving the equation $r = DT(\Phi r)$. This algorithm contains as a special case the iterative aggregation algorithm (6.169), as well as other algorithms of interest in DP, such as for example Q-learning and aggregation-type algorithms for stochastic shortest path problems, and for problems involving post-decision states.

As in Q-learning, the starting point of the algorithm is the fixed point iteration

$$r_{k+1} = DT(\Phi r_k).$$

This iteration is guaranteed to converge to r^*, and the same is true for asynchronous versions where only one component of r is updated at each iteration (this is due to the sup-norm contraction property of $DT\Phi$). To obtain a simulation-based approximation of DT, we introduce an $s \times m$ matrix $\overline{D}$ whose rows are m-dimensional probability distributions with components $\overline{d}_{\ell i}$ satisfying

$$\overline{d}_{\ell i} > 0 \qquad \text{if } d_{\ell i} \neq 0, \qquad \ell = 1, \ldots, s,\ i = 1, \ldots, m.$$

The ℓth component of the vector $DT(\Phi r)$ can be written as an expected value with respect to this distribution:

$$\sum_{i=1}^{m} d_{\ell i} T_i(\Phi r) = \sum_{i=1}^{m} \overline{d}_{\ell i} \left(\frac{d_{\ell i}}{\overline{d}_{\ell i}}\, T_i(\Phi r) \right), \tag{6.287}$$

where T_i is the ith component of T. This expected value is approximated by simulation in the algorithm that follows.

The algorithm generates a sequence of indices $\{\ell_0, \ell_1, \ldots\}$ according to some mechanism that ensures that all indices $\ell = 1, \ldots, s$, are generated infinitely often. Given ℓ_k, an index $i_k \in \{1, \ldots, m\}$ is generated according to the probabilities $\overline{d}_{\ell_k i}$, independently of preceding indices. Then the components of r_k, denoted $r_k(\ell)$, $\ell = 1, \ldots, s$, are updated using the following iteration:

$$r_{k+1}(\ell) = \begin{cases} (1 - \gamma_k) r_k(\ell) + \gamma_k \dfrac{d_{\ell i_k}}{\overline{d}_{\ell i_k}}\, T_{i_k}(\Phi r_k) & \text{if } \ell = \ell_k, \\ r_k(\ell) & \text{if } \ell \neq \ell_k, \end{cases}$$

where $\gamma_k > 0$ is a stepsize that diminishes to 0 at an appropriate rate. Thus only the ℓ_kth component of r_k is changed, while all other components are left unchanged. The stepsize could be chosen to be $\gamma_k = 1/n_k$, where as in

Section 6.5.1, n_k is the number of times that index ℓ_k has been generated within the sequence $\{\ell_0, \ell_1, \ldots\}$ up to time k.

The algorithm is similar and indeed contains as a special case the Q-learning algorithm (6.181)-(6.182). The justification of the algorithm follows closely the one given for Q-learning in Section 6.5.1. Basically, we replace the expected value in the expression (6.287) of the ℓth component of DT, with a Monte Carlo estimate based on all the samples up to time k that involve ℓ_k, and we then simplify the hard-to-calculate terms in the resulting method [cf. Eqs. (6.190) and (6.192)]. A rigorous convergence proof requires the theoretical machinery of stochastic approximation algorithms.

6.9 APPROXIMATION IN POLICY SPACE

Our approach so far in this chapter has been to use an approximation architecture for some cost function, differential cost, or Q-factor. Sometimes this is called *approximation in value space*, to indicate that a cost or value function is being approximated. In an important alternative, called *approximation in policy space*, we parameterize the set of policies by a vector $r = (r_1, \ldots, r_s)$ and we optimize the cost over this vector. In particular, we consider randomized stationary policies of a given parametric form $\tilde{\mu}_u(i, r)$, where $\tilde{\mu}_u(i, r)$ denotes the probability that control u is applied when the state is i. Each value of r defines a randomized stationary policy, which in turn defines the cost of interest as a function of r. We then choose r to minimize this cost.

In an important special case of this approach, the parameterization of the policies is indirect, through an approximate cost function. In particular, a cost approximation architecture parameterized by r, defines a policy dependent on r via the minimization in Bellman's equation. For example, Q-factor approximations $\tilde{Q}(i, u, r)$, define a parameterization of policies by letting $\tilde{\mu}_u(i, r) = 1$ for some u that minimizes $\tilde{Q}(i, u, r)$ over $u \in U(i)$, and $\tilde{\mu}_u(i, r) = 0$ for all other u. This parameterization is discontinuous in r, but in practice is it smoothed by replacing the minimization operation with a smooth exponential-based approximation; we refer to the literature for the details. Also in a more abstract and general view of approximation in policy space, rather than parameterizing policies or Q-factors, we can simply parameterize by r the problem data (stage costs and transition probabilities), and optimize the corresponding cost function over r. Thus, in this more general formulation, we may aim to select some parameters of a given system to optimize performance.

Once policies are parameterized in some way by a vector r, the cost function of the problem, over a finite or infinite horizon, is implicitly parameterized as a vector $\tilde{J}(r)$. A scalar measure of performance may then be derived from $\tilde{J}(r)$, e.g., the expected cost starting from a single initial

state, or a weighted sum of costs starting from a selected set of states. The method of optimization may be any one of a number of possible choices, ranging from random search to gradient methods. This method need not relate to DP, although DP calculations may play a significant role in its implementation. Traditionally, gradient-type methods have received most attention within this context, but they often tend to be slow and to have difficulties with local minima. On the other hand, random search methods, such as the cross-entropy method [RuK04], are often very easy to implement and on occasion have proved surprisingly effective (see the literature cited in Section 6.10).

In this section, we will focus on the finite spaces average cost problem and gradient-type methods. Let the cost per stage vector and transition probability matrix be given as functions of r: $G(r)$ and $P(r)$, respectively. Assume that the states form a single recurrent class under each $P(r)$, and let $\xi(r)$ be the corresponding steady-state probability vector. We denote by $G_i(r)$, $P_{ij}(r)$, and $\xi_i(r)$ the components of $G(r)$, $P(r)$, and $\xi(r)$, respectively. Each value of r defines an average cost $\eta(r)$, which is common for all initial states (cf. Section 4.2), and the problem is to find

$$\min_{r \in \Re^s} \eta(r).$$

Assuming that $\eta(r)$ is differentiable with respect to r (something that must be independently verified), one may use a gradient method for this minimization:

$$r_{k+1} = r_k - \gamma_k \nabla \eta(r_k),$$

where γ_k is a positive stepsize. This is known as a *policy gradient method*.

6.9.1 The Gradient Formula

We will now show that a convenient formula for the gradients $\nabla \eta(r)$ can be obtained by differentiating Bellman's equation

$$\eta(r) + h_i(r) = G_i(r) + \sum_{j=1}^{n} P_{ij}(r) h_j(r), \qquad i = 1, \ldots, n, \tag{6.288}$$

with respect to the components of r, where $h_i(r)$ are the differential costs. Taking the partial derivative with respect to r_m, we obtain for all i and m,

$$\frac{\partial \eta}{\partial r_m} + \frac{\partial h_i}{\partial r_m} = \frac{\partial G_i}{\partial r_m} + \sum_{j=1}^{n} \frac{\partial P_{ij}}{\partial r_m} h_j + \sum_{j=1}^{n} P_{ij} \frac{\partial h_j}{\partial r_m}.$$

(In what follows we assume that the partial derivatives with respect to components of r appearing in various equations exist. The argument at which they are evaluated, is often suppressed to simplify notation.) By

multiplying this equation with $\xi_i(r)$, adding over i, and using the fact $\sum_{i=1}^n \xi_i(r) = 1$, we obtain

$$\frac{\partial \eta}{\partial r_m} + \sum_{i=1}^n \xi_i \frac{\partial h_i}{\partial r_m} = \sum_{i=1}^n \xi_i \frac{\partial G_i}{\partial r_m} + \sum_{i=1}^n \xi_i \sum_{j=1}^n \frac{\partial P_{ij}}{\partial r_m} h_j + \sum_{i=1}^n \xi_i \sum_{j=1}^n P_{ij} \frac{\partial h_j}{\partial r_m}.$$

The last summation on the right-hand side cancels the last summation on the left-hand side, because from the defining property of the steady-state probabilities, we have

$$\sum_{i=1}^n \xi_i \sum_{j=1}^n P_{ij} \frac{\partial h_j}{\partial r_m} = \sum_{j=1}^n \left(\sum_{i=1}^n \xi_i P_{ij} \right) \frac{\partial h_j}{\partial r_m} = \sum_{j=1}^n \xi_j \frac{\partial h_j}{\partial r_m}.$$

We thus obtain

$$\frac{\partial \eta(r)}{\partial r_m} = \sum_{i=1}^n \xi_i(r) \left(\frac{\partial G_i(r)}{\partial r_m} + \sum_{j=1}^n \frac{\partial P_{ij}(r)}{\partial r_m} h_j(r) \right), \qquad m = 1, \ldots, s, \tag{6.289}$$

or in more compact form

$$\nabla \eta(r) = \sum_{i=1}^n \xi_i(r) \left(\nabla G_i(r) + \sum_{j=1}^n \nabla P_{ij}(r) h_j(r) \right), \tag{6.290}$$

where all the gradients are column vectors of dimension s.

6.9.2 Computing the Gradient by Simulation

Despite its relative simplicity, the gradient formula (6.290) involves formidable computations to obtain $\nabla \eta(r)$ at just a single value of r. The reason is that neither the steady-state probability vector $\xi(r)$ nor the bias vector $h(r)$ are readily available, so they must be computed or approximated in some way. Furthermore, $h(r)$ is a vector of dimension n, so for large n, it can only be approximated either through its simulation samples or by using a parametric architecture and an algorithm such as LSPE or LSTG (see the references cited at the end of the chapter).

The possibility to approximate h using a parametric architecture ushers a connection between approximation in policy space and approximation in value space. It also raises the question whether approximations introduced in the gradient calculation may affect the convergence guarantees of the policy gradient method. Fortunately, however, gradient algorithms tend to be robust and maintain their convergence properties, even in the presence of significant error in the calculation of the gradient.

In the literature, algorithms where both μ and h are parameterized are sometimes called *actor-critic* methods. Algorithms where just μ is parameterized and h is not parameterized but rather estimated explicitly or implicitly by simulation, are called *actor-only* methods, while algorithms where just h is parameterized and μ is obtained by one-step lookahead minimization, are called *critic-only* methods.

We will now discuss some possibilities of using simulation to approximate $\nabla\eta(r)$. Let us introduce for all i and j such that $P_{ij}(r) > 0$, the function

$$L_{ij}(r) = \frac{\nabla P_{ij}(r)}{P_{ij}(r)}.$$

Then, suppressing the dependence on r, we write the partial derivative formula (6.290) in the form

$$\nabla\eta = \sum_{i=1}^{n} \xi_i \left(\nabla G_i + \sum_{j=1}^{n} P_{ij} L_{ij} h_j \right). \tag{6.291}$$

We assume that for all states i and possible transitions (i,j), we can calculate ∇G_i and L_{ij}. Suppose now that we generate a single infinitely long simulated trajectory $(i_0, i_1, \ldots)$. We can then estimate the average cost η as

$$\tilde{\eta} = \frac{1}{k} \sum_{t=0}^{k-1} G_{i_t},$$

where k is large. Then, given an estimate $\tilde{\eta}$, we can estimate the bias components h_j by using simulation-based approximations to the formula

$$h_{i_0} = \lim_{N\to\infty} E\left\{ \sum_{t=0}^{N} (G_{i_t} - \eta) \right\},$$

[which holds from general properties of the bias vector when $P(r)$ is aperiodic – see the discussion following Prop. 4.1.2]. Alternatively, we can estimate h_j by using the LSPE or LSTD algorithms of Section 6.7.1 [note here that if the feature subspace contains the bias vector, the LSPE and LSTD algorithms will find exact values of h_j in the limit, so with a sufficiently rich set of features, an asymptotically exact calculation of h_j, and hence also $\nabla\eta(r)$, is possible]. Finally, given estimates $\tilde{\eta}$ and $\tilde{h}_j$, we can estimate the gradient $\nabla\eta$ with a vector δ_η given by

$$\delta_\eta = \frac{1}{k} \sum_{t=0}^{k-1} \big(\nabla G_{i_t} + L_{i_t i_{t+1}} \tilde{h}_{i_{t+1}} \big). \tag{6.292}$$

This can be seen by a comparison of Eqs. (6.291) and (6.292): if we replace the expected values of ∇G_i and L_{ij} by empirical averages, and we replace h_j by $\tilde{h}_j$, we obtain the estimate δ_η.

The estimation-by-simulation procedure outlined above provides a conceptual starting point for more practical gradient estimation methods. For example, in such methods, the estimation of η and h_j may be done simultaneously with the estimation of the gradient via Eq. (6.292), and with a variety of different algorithms. We refer to the literature cited at the end of the chapter.

6.9.3 Essential Features of Critics

We will now develop an alternative (but mathematically equivalent) expression for the gradient $\nabla\eta(r)$ that involves Q-factors instead of differential costs. Let us consider randomized policies where $\tilde{\mu}_u(i,r)$ denotes the probability that control u is applied at state i. We assume that $\tilde{\mu}_u(i,r)$ is differentiable with respect to r for each i and u. Then the corresponding stage costs and transition probabilities are given by

$$G_i(r) = \sum_{u\in U(i)} \tilde{\mu}_u(i,r) \sum_{j=1}^{n} p_{ij}(u)g(i,u,j), \qquad i = 1,\ldots,n,$$

$$P_{ij}(r) = \sum_{u\in U(i)} \tilde{\mu}_u(i,r)p_{ij}(u), \qquad i,j = 1,\ldots,n.$$

Differentiating these equations with respect to r, we obtain

$$\nabla G_i(r) = \sum_{u\in U(i)} \nabla\tilde{\mu}_u(i,r) \sum_{j=1}^{n} p_{ij}(u)g(i,u,j), \tag{6.293}$$

$$\nabla P_{ij}(r) = \sum_{u\in U(i)} \nabla\tilde{\mu}_u(i,r)p_{ij}(u), \qquad i,j = 1,\ldots,n. \tag{6.294}$$

Since $\sum_{u\in U(i)} \tilde{\mu}_u(i,r) = 1$ for all r, we have $\sum_{u\in U(i)} \nabla\tilde{\mu}_u(i,r) = 0$, so Eq. (6.293) yields

$$\nabla G_i(r) = \sum_{u\in U(i)} \nabla\tilde{\mu}_u(i,r) \left(\sum_{j=1}^{n} p_{ij}(u)g(i,u,j) - \eta(r) \right).$$

Also, by multiplying with $h_j(r)$ and adding over j, Eq. (6.294) yields

$$\sum_{j=1}^{n} \nabla P_{ij}(r)h_j(r) = \sum_{j=1}^{n} \sum_{u\in U(i)} \nabla\tilde{\mu}_u(i,r)p_{ij}(u)h_j(r).$$

By using the preceding two equations to rewrite the gradient formula (6.290), we obtain

$$\begin{aligned}\nabla\eta(r) &= \sum_{i=1}^{n}\xi_i(r)\left(\nabla G_i(r)+\sum_{j=1}^{n}\nabla P_{ij}(r)h_j(r)\right)\\ &= \sum_{i=1}^{n}\xi_i(r)\sum_{u\in U(i)}\nabla\tilde{\mu}_u(i,r)\sum_{j=1}^{n}p_{ij}(u)\big(g(i,u,j)-\eta(r)+h_j(r)\big),\end{aligned}$$

and finally

$$\nabla\eta(r)=\sum_{i=1}^{n}\sum_{u\in U(i)}\xi_i(r)\tilde{Q}(i,u,r)\nabla\tilde{\mu}_u(i,r), \tag{6.295}$$

where $\tilde{Q}(i,u,r)$ are the approximate Q-factors corresponding to r:

$$\tilde{Q}(i,u,r)=\sum_{j=1}^{n}p_{ij}(u)\big(g(i,u,j)-\eta(r)+h_j(r)\big).$$

Let us now express the formula (6.295) in a way that is amenable to proper interpretation. In particular, by writing

$$\nabla\eta(r)=\sum_{i=1}^{n}\sum_{\{u\in U(i)\mid\tilde{\mu}_u(i,r)>0\}}\xi_i(r)\tilde{\mu}_u(i,r)\tilde{Q}(i,u,r)\frac{\nabla\tilde{\mu}_u(i,r)}{\tilde{\mu}_u(i,r)},$$

and by introducing the function

$$\psi_r(i,u)=\frac{\nabla\tilde{\mu}_u(i,r)}{\tilde{\mu}_u(i,r)},$$

we obtain

$$\nabla\eta(r)=\sum_{i=1}^{n}\sum_{\{u\in U(i)\mid\tilde{\mu}_u(i,r)>0\}}\zeta_r(i,u)\tilde{Q}(i,u,r)\psi_r(i,u), \tag{6.296}$$

where $\zeta_r(i,u)$ are the steady-state probabilities of the pairs (i,u) under r:

$$\zeta_r(i,u)=\xi_i(r)\tilde{\mu}_u(i,r).$$

Note that for each (i,u), $\psi_r(i,u)$ is a vector of dimension s, the dimension of the parameter vector r. We denote by $\psi_r^m(i,u)$, $m=1,\ldots,s$, the components of this vector.

Equation (6.296) can form the basis for policy gradient methods that estimate $\tilde{Q}(i,u,r)$ by simulation, thereby leading to actor-only algorithms. An alternative suggested by Konda and Tsitsiklis [KoT99], [KoT03], is to

interpret the formula as an inner product, thereby leading to a different set of algorithms. In particular, for a given r, we define the inner product of two real-valued functions Q_1, Q_2 of (i, u), by

$$\langle Q_1, Q_2 \rangle_r = \sum_{i=1}^{n} \sum_{\{u \in U(i) | \tilde{\mu}_u(i,r) > 0\}} \zeta_r(i,u) Q_1(i,u) Q_2(i,u).$$

With this notation, we can rewrite Eq. (6.296) as

$$\frac{\partial \eta(r)}{\partial r_m} = \langle \tilde{Q}(\cdot, \cdot, r), \psi_r^m(\cdot, \cdot) \rangle_r, \qquad m = 1, \ldots, s.$$

An important observation is that although $\nabla \eta(r)$ depends on $\tilde{Q}(i, u, r)$, which has a number of components equal to the number of state-control pairs (i, u), the dependence is only through its inner products with the s functions $\psi_r^m(\cdot, \cdot)$, $m = 1, \ldots, s$.

Now let $\| \cdot \|_r$ be the norm induced by this inner product, i.e.,

$$\|Q\|_r^2 = \langle Q, Q \rangle_r.$$

Let also S_r be the subspace that is spanned by the functions $\psi_r^m(\cdot, \cdot)$, $m = 1, \ldots, s$, and let Π_r denote projection with respect to this norm onto S_r. Since

$$\langle \tilde{Q}(\cdot, \cdot, r), \psi_r^m(\cdot, \cdot) \rangle_r = \langle \Pi_r \tilde{Q}(\cdot, \cdot, r), \psi_r^m(\cdot, \cdot) \rangle_r, \qquad m = 1, \ldots, s,$$

it is sufficient to know the projection of $\tilde{Q}(\cdot, \cdot, r)$ onto S_r in order to compute $\nabla \eta(r)$. Thus S_r defines a subspace of *essential features*, i.e., features the knowledge of which is essential for the calculation of the gradient $\nabla \eta(r)$. As discussed in Section 6.1, the projection of $\tilde{Q}(\cdot, \cdot, r)$ onto S_r can be done in an approximate sense with TD(λ), LSPE(λ), or LSTD(λ) for $\lambda \approx 1$. We refer to the papers by Konda and Tsitsiklis [KoT99], [KoT03], and Sutton, McAllester, Singh, and Mansour [SMS99] for further discussion.

6.9.4 Approximations in Policy and Value Space

Let us now provide a comparative assessment of approximation in policy and value space. We first note that in comparing approaches, one must bear in mind that specific problems may admit natural parametrizations that favor one type of approximation over the other. For example, in inventory control problems, it is natural to consider policy parametrizations that resemble the (s, S) policies that are optimal for special cases, but also make intuitive sense in a broader context.

Policy gradient methods for approximation in policy space are supported by interesting theory and aim directly at finding an optimal policy

within the given parametric class (as opposed to aiming for policy evaluation in the context of an approximate policy iteration scheme). However, they suffer from a drawback that is well-known to practitioners of nonlinear optimization: slow convergence, which unless improved through the use of effective scaling of the gradient (with an appropriate diagonal or nondiagonal matrix), all too often leads to jamming (no visible progress) and complete breakdown. Unfortunately, there has been no proposal of a demonstrably effective scheme to scale the gradient in policy gradient methods (see, however, Kakade [Kak02] for an interesting attempt to address this issue, based on the work of Amari [Ama98]). Furthermore, the performance and reliability of policy gradient methods are susceptible to degradation by large variance of simulation noise. Thus, while policy gradient methods are supported by convergence guarantees in theory, attaining convergence in practice is often challenging. In addition, gradient methods have a generic difficulty with local minima, the consequences of which are not well-understood at present in the context of approximation in policy space.

A major difficulty for approximation in value space is that a good choice of basis functions/features is often far from evident. Furthermore, even when good features are available, the indirect approach of TD(λ), LSPE(λ), and LSTD(λ) may neither yield the best possible approximation of the cost function or the Q-factors of a policy within the feature subspace, nor yield the best possible performance of the associated one-step-lookahead policy. In the case of a fixed policy, LSTD(λ) and LSPE(λ) are quite reliable algorithms, in the sense that they ordinarily achieve their theoretical guarantees in approximating the associated cost function or Q-factors: they involve solution of systems of linear equations, simulation (with convergence governed by the law of large numbers), and contraction iterations (with favorable contraction modulus when λ is not too close to 0). However, within the multiple policy context of an approximate policy iteration scheme, TD methods have additional difficulties: the need for adequate exploration, the issue of policy oscillation and the related chattering phenomenon, and the lack of convergence guarantees for both optimistic and nonoptimistic schemes. When an aggregation method is used for policy evaluation, these difficulties do not arise, but the cost approximation vectors Φr are restricted by the requirements that the rows of Φ must be aggregation probability distributions.

6.10 NOTES, SOURCES, AND EXERCISES

There has been intensive interest in simulation-based methods for approximate DP since the early 90s, in view of their promise to address the dual curses of DP: the curse of dimensionality (the explosion of the computa-

tion needed to solve the problem as the number of states increases), and the curse of modeling (the need for an exact model of the system's dynamics). We have used the name *approximate dynamic programming* to collectively refer to these methods. Two other popular names are *reinforcement learning* and *neuro-dynamic programming*. The latter name, adopted by Bertsekas and Tsitsiklis [BeT96], comes from the strong connections with DP as well as with methods traditionally developed in the field of neural networks, such as the training of approximation architectures using empirical or simulation data.

Two books were written on the subject in the mid-90s, one by Sutton and Barto [SuB98], which reflects an artificial intelligence viewpoint, and another by Bertsekas and Tsitsiklis [BeT96], which is more mathematical and reflects an optimal control/operations research viewpoint. We refer to the latter book for a broader discussion of some of the topics of this chapter [including rigorous convergence proofs of TD(λ) and Q-learning], for related material on approximation architectures, batch and incremental gradient methods, and neural network training, as well as for an extensive overview of the history and bibliography of the subject up to 1996. More recent books are Cao [Cao07], which emphasizes a sensitivity approach and policy gradient methods, Chang, Fu, Hu, and Marcus [CFH07], which emphasizes finite-horizon/limited lookahead schemes and adaptive sampling, Gosavi [Gos03], which emphasizes simulation-based optimization and reinforcement learning algorithms, Powell [Pow07], which emphasizes resource allocation and the difficulties associated with large control spaces, and Busoniu et. al. [BBD10], which focuses on function approximation methods for continuous space systems. The book by Haykin [Hay08] discusses approximate DP within the broader context of neural networks and learning. The book by Borkar [Bor08] is an advanced monograph that addresses rigorously many of the convergence issues of iterative stochastic algorithms in approximate DP, mainly using the so called ODE approach (see also Borkar and Meyn [BoM00]). The book by Meyn [Mey07] is broader in its coverage, but touches upon some of the approximate DP algorithms that we have discussed.

Several survey papers in the volume by Si, Barto, Powell, and Wunsch [SBP04], and the special issue by Lewis, Liu, and Lendaris [LLL08] describe recent work and approximation methodology that we have not covered in this chapter: linear programming-based approaches (De Farias and Van Roy [DFV03], [DFV04a], De Farias [DeF04]), large-scale resource allocation methods (Powell and Van Roy [PoV04]), and deterministic optimal control approaches (Ferrari and Stengel [FeS04], and Si, Yang, and Liu [SYL04]). An influential survey was written, from an artificial intelligence/machine learning viewpoint, by Barto, Bradtke, and Singh [BBS95]. Some recent surveys are Borkar [Bor09] (a methodological point of view that explores connections with other Monte Carlo schemes), Lewis and Vrabie [LeV09] (a control theory point of view), and Szepesvari [Sze09] (a

machine learning point of view), Bertsekas [Ber10a] (which focuses on rollout algorithms for discrete optimization), and Bertsekas [Ber10b] (which focuses on policy iteration and elaborates on some of the topics of this chapter). The reader is referred to these sources for a broader survey of the literature of approximate DP, which is very extensive and cannot be fully covered here.

Direct approximation methods and the fitted value iteration approach have been used for finite horizon problems since the early days of DP. They are conceptually simple and easily implementable, and they are still in wide use for approximation of either optimal cost functions or Q-factors (see e.g., Gordon [Gor99], Longstaff and Schwartz [LoS01], Ormoneit and Sen [OrS02], and Ernst, Geurts, and Wehenkel [EGW06]). The simplifications mentioned in Section 6.1.4 are part of the folklore of DP. In particular, post-decision states have sporadically appeared in the literature since the early days of DP. They were used in an approximate DP context by Van Roy, Bertsekas, Lee, and Tsitsiklis [VBL97] in the context of inventory control problems. They have been recognized as an important simplification in the book by Powell [Pow07], which pays special attention to the difficulties associated with large control spaces. For a recent application, see Simao et. al. [SDG09].

Temporal differences originated in reinforcement learning, where they are viewed as a means to encode the error in predicting future costs, which is associated with an approximation architecture. They were introduced in the works of Samuel [Sam59], [Sam67] on a checkers-playing program. The papers by Barto, Sutton, and Anderson [BSA83], and Sutton [Sut88] proposed the TD(λ) method, on a heuristic basis without a convergence analysis. The method motivated a lot of research in simulation-based DP, particularly following an early success with the backgammon playing program of Tesauro [Tes92]. The original papers did not discuss mathematical convergence issues and did not make the connection of TD methods with the projected equation. Indeed for quite a long time it was not clear which mathematical problem TD(λ) was aiming to solve! The convergence of TD(λ) and related methods was considered for discounted problems by several authors, including Dayan [Day92], Gurvits, Lin, and Hanson [GLH94], Jaakkola, Jordan, and Singh [JJS94], Pineda [Pin97], Tsitsiklis and Van Roy [TsV97], and Van Roy [Van98]. The proof of Tsitsiklis and Van Roy [TsV97] was based on the contraction property of ΠT (cf. Lemma 6.3.1 and Prop. 6.3.1), which is the starting point of our analysis of Section 6.3. The scaled version of TD(0) [cf. Eq. (6.80)] as well as a λ-counterpart were proposed by Choi and Van Roy [ChV06] under the name Fixed Point Kalman Filter. The books by Bertsekas and Tsitsiklis [BeT96], and Sutton and Barto [SuB98] contain a lot of material on TD(λ), its variations, and its use in approximate policy iteration.

Generally, projected equations are the basis for Galerkin methods, which are popular in scientific computation (see e.g., [Kra72], [Fle84]).

These methods typically do not use Monte Carlo simulation, which is essential for the DP context. However, Galerkin methods apply to a broad range of problems, far beyond DP, which is in part the motivation for our discussion of projected equations in more generality in Section 6.8.

The LSTD(λ) algorithm was first proposed by Bradtke and Barto [BrB96] for $\lambda = 0$, and later extended by Boyan [Boy02] for $\lambda > 0$. For $\lambda > 0$, the convergence $C_k^{(\lambda)} \to C^{(\lambda)}$ and $d_k^{(\lambda)} \to d^{(\lambda)}$ is not as easy to demonstrate as in the case $\lambda = 0$. An analysis of the law-of-large-numbers convergence issues associated with LSTD for discounted problems was given by Nedić and Bertsekas [NeB03]. The more general two-Markov chain sampling context that can be used for exploration-related methods is analyzed by Bertsekas and Yu [BeY09], and by Yu [Yu10a,b], which shows convergence under the most general conditions. The analysis of [BeY09] and [Yu10a,b] also extends to simulation-based solution of general projected equations. The rate of convergence of LSTD was analyzed by Konda [Kon02], who showed that LSTD has optimal rate of convergence within a broad class of temporal difference methods. The regression/regularization variant of LSTD is due to Wang, Polydorides, and Bertsekas [WPB09]. This work addresses more generally the simulation-based approximate solution of linear systems and least squares problems, and it applies to LSTD as well as to the minimization of the Bellman equation error as special cases.

The LSPE(λ) algorithm, was first proposed for stochastic shortest path problems by Bertsekas and Ioffe [BeI96], and was applied to a challenging problem on which TD(λ) failed: learning an optimal strategy to play the game of tetris (see also Bertsekas and Tsitsiklis [BeT96], Section 8.3). The convergence of the method for discounted problems was given in [NeB03] (for a diminishing stepsize), and by Bertsekas, Borkar, and Nedić [BBN04] (for a unit stepsize). In the paper [BeI96] and the book [BeT96], the LSPE method was related to the λ-policy iteration of Section 6.3.9. The paper [BBN04] compared informally LSPE and LSTD for discounted problems, and suggested that they asymptotically coincide in the sense described in Section 6.3. Yu and Bertsekas [YuB06b] provided a mathematical proof of this for both discounted and average cost problems. The scaled versions of LSPE and the associated convergence analysis were developed more recently, and within a more general context in Bertsekas [Ber09b], [Ber11a], which are based on a connection between general projected equations and variational inequalities. Some other iterative methods were given by Yao and Liu [YaL08]. The research on policy or Q-factor evaluation methods was of course motivated by their use in approximate policy iteration schemes. There has been considerable experimentation with such schemes, see e.g., [BeI96], [BeT96], [SuB98], [LaP03], [JuP07], [BED09]. However, the relative practical advantages of optimistic versus nonoptimistic schemes, in conjunction with LSTD, LSPE, and TD(λ), are not yet clear. The exploration-enhanced versions of LSPE(λ) and LSTD(λ) of Section 6.3.6 are new and were developed as alternative implementations

of the λ-policy iteration method [Ber11b].

Policy oscillations and chattering were first described by the author at an April 1996 workshop on reinforcement learning [Ber96], and were subsequently discussed in Section 6.4.2 of [BeT96]. The size of the oscillations is bounded by the error bound of Prop. 1.3.6, which is due to [BeT96]. An alternative error bound that is based on the Euclidean norm has been derived by Munos [Mun03], and by Scherrer [Sch07] who considered the λ-policy iteration algorithm of Section 6.3.9. Feature scaling and its effect on LSTD(λ), LSPE(λ), and TD(λ) (Section 6.3.6) was discussed in Bertsekas [Ber11a]. The conditions for policy convergence of Section 6.3.8 were derived in Bertsekas [Ber10b] and [Ber10c].

The exploration scheme with extra transitions (Section 6.3.7) was given in the paper by Bertsekas and Yu [BeY09], Example 1. The LSTD(λ) algorithm with exploration and modified temporal differences (Section 6.3.7) was given by Bertsekas and Yu [BeY07], and a convergence with probability 1 proof was provided under the condition $\lambda\alpha p_{ij} \leq \overline{p}_{ij}$ for all (i,j) in [BeY09], Prop. 4. The idea of modified temporal differences stems from the techniques of importance sampling, which have been introduced in various DP-related contexts by a number of authors: Glynn and Iglehart [GlI89] (for exact cost evaluation), Precup, Sutton, and Dasgupta [PSD01] [for TD(λ) with exploration and stochastic shortest path problems], Ahamed, Borkar, and Juneja [ABJ06] (in adaptive importance sampling schemes for cost vector estimation without approximation), and Bertsekas and Yu [BeY07], [BeY09] (in the context of the generalized projected equation methods of Section 6.8.1).

The λ-policy iteration algorithm discussed in Section 6.3.9 was first proposed by Bertsekas and Ioffe [BeI96], and it was used as the basis for the original development of LSPE and its application to the tetris problem (see also [BeT96], Sections 2.3.1 and 8.3). The name "LSPE" was first used in the subsequent paper by Nedić and Bertsekas [NeB03] to describe a specific iterative implementation of the λ-PI method with cost function approximation for discounted MDP (essentially the implementation developed in [BeI96] and [BeT96], and used for a tetris case study). The second simulation-based implementation described in this section, which views the policy evaluation problem in the context of a stopping problem, is new (see Bertsekas [Ber11b]). The third simulation-based implementation in this section, was proposed by Thierry and Scherrer [ThS10a], [ThS10b], who proposed various associated optimistic policy iteration implementations that relate to both LSPE and LSTD.

The aggregation approach has a long history in scientific computation and operations research. It was introduced in the simulation-based approximate DP context, mostly in the form of value iteration; see Singh, Jaakkola, and Jordan [SJJ94], [SJJ95], Gordon [Gor95], Tsitsiklis and Van Roy [TsV96], and Van Roy [Van06]. Bounds on the error between the optimal cost-to-go vector J^* and the limit of the value iteration method in the

case of hard aggregation are given under various assumptions in [TsV96] (see also Exercise 6.12 and Section 6.7.4 of [BeT96]). Related error bounds are given by Munos and Szepesvari [MuS08]. A more recent work that focuses on hard aggregation is Van Roy [Van06]. The analysis given here, which follows the lines of Section 6.3.4 of Vol. I and emphasizes the importance of convergence in approximate policy iteration, is somewhat different from alternative developments in the literature.

Multistep aggregation does not seem to have been considered in the literature, but it may have some important practical applications in problems where multistep lookahead minimizations are feasible. Also asynchronous distributed aggregation has not been discussed earlier. It is worth emphasizing that while both projected equation and aggregation methods produce basis function approximations to costs or Q-factors, there is an important qualitative difference that distinguishes the aggregation-based policy iteration approach: assuming sufficiently small simulation error, it is not susceptible to policy oscillation and chattering like the projected equation or Bellman equation error approaches. The price for this is the restriction of the type of basis functions that can be used in aggregation.

Q-learning was proposed by Watkins [Wat89], who explained the essence of the method, but did not provide a rigorous convergence analysis; see also Watkins and Dayan [WaD92]. A convergence proof was given by Tsitsiklis [Tsi94]. For SSP problems with improper policies, this proof required the assumption of nonnegative one-stage costs (see also [BeT96], Prop. 5.6). This assumption was relaxed by Abounadi, Bertsekas, and Borkar [ABB02], under some conditions and using an alternative line of proof, based on the so-called ODE approach. The proofs of these references include the assumption that either the iterates are bounded or other related restrictions. It was shown by Yu and Bertsekas [YuB11] that the Q-learning iterates are naturally bounded for SSP problems, even with improper policies, so the convergence of Q-learning for SSP problems was established under no more restrictive assumptions than for discounted MDP.

A variant of Q-learning is the method of *advantage updating*, developed by Baird [Bai93], [Bai94], [Bai95], and Harmon, Baird, and Klopf [HBK94]. In this method, instead of aiming to compute $Q(i,u)$, we compute

$$A(i,u) = Q(i,u) - \min_{u \in U(i)} Q(i,u).$$

The function $A(i,u)$ can serve just as well as $Q(i,u)$ for the purpose of computing corresponding policies, based on the minimization $\min_{u \in U(i)} A(i,u)$, but may have a much smaller range of values than $Q(i,u)$, which may be helpful in contexts involving basis function approximation. When using a lookup table representation, advantage updating is essentially equivalent to Q-learning, and has the same type of convergence properties. With function approximation, the convergence properties of advantage updating are

not well-understood (similar to Q-learning). We refer to the book [BeT96], Section 6.6.2, for more details and some analysis.

Another variant of Q-learning, also motivated by the fact that we are really interested in Q-factor differences rather than Q-factors, has been discussed in Section 6.4.2 of Vol. I, and is aimed at variance reduction of Q-factors obtained by simulation. A related variant of approximate policy iteration and Q-learning, called *differential training*, has been proposed by the author in [Ber97] (see also Weaver and Baxter [WeB99]). It aims to compute Q-factor differences in the spirit of the variance reduction ideas of Section 6.4.2 of Vol. I.

Approximation methods for the optimal stopping problem (Section 6.5.3) were investigated by Tsitsiklis and Van Roy [TsV99b], [Van98], who noted that Q-learning with a linear parametric architecture could be applied because the associated mapping F is a contraction with respect to the norm $\|\cdot\|_\xi$. They proved the convergence of a corresponding Q-learning method, and they applied it to a problem of pricing financial derivatives. The LSPE algorithm given in Section 6.5.3 for this problem is due to Yu and Bertsekas [YuB07], to which we refer for additional analysis. An alternative algorithm with some similarity to LSPE as well as TD(0) is given by Choi and Van Roy [ChV06], and is also applied to the optimal stopping problem. We note that approximate dynamic programming and simulation methods for stopping problems have become popular in the finance area, within the context of pricing options; see Longstaff and Schwartz [LoS01], who consider a finite horizon model in the spirit of Section 6.5.4, and Tsitsiklis and Van Roy [TsV01], and Li, Szepesvari, and Schuurmans [LSS09], whose works relate to the LSPE method of Section 6.5.3. The constrained policy iteration method of Section 6.5.3 is closely related to the paper by Bertsekas and Yu [BeY10a].

Recently, an approach to Q-learning with exploration, called *enhanced policy iteration*, has been proposed (Bertsekas and Yu [BeY10a]). Instead of policy evaluation by solving a linear system of equations, this method requires (possibly inexact) solution of Bellman's equation for an optimal stopping problem. It is based on replacing the standard Q-learning mapping used for evaluation of a policy μ with the mapping

$$(F_{J,\nu}Q)(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \sum_{v \in U(j)} \nu(v \mid j) \min\bigl\{J(j), Q(j,v)\bigr\} \right)$$

which depends on a vector $J \in \Re^n$, with components denoted $J(i)$, and on a randomized policy ν, which for each state i defines a probability distribution

$$\bigl\{\nu(u \mid i) \mid u \in U(i)\bigr\}$$

over the feasible controls at i, and may depend on the "current policy" μ. The vector J is updated using the equation $J(i) = \min_{u \in U(i)} Q(i,u)$, and

the "current policy" μ is obtained from this minimization. Finding a fixed point of the mapping $F_{J,\nu}$ is an optimal stopping problem [a similarity with the constrained policy iteration (6.210)-(6.211)]. The policy ν may be chosen arbitrarily at each iteration. It encodes aspects of the "current policy" μ, but allows for arbitrary and easily controllable amount of exploration. For extreme choices of ν and a lookup table representation, the algorithms of [BeY10a] yield as special cases the classical Q-learning/value iteration and policy iteration methods. Together with linear cost/Q-factor approximation, the algorithms may be combined with the TD(0)-like method of Tsitsiklis and Van Roy [TsV99b], which can be used to solve the associated stopping problems with low overhead per iteration, thereby resolving the issue of exploration. Reference [BeY10a] also provides optimistic asynchronous policy iteration versions of Q-learning, which have guaranteed convergence properties and lower overhead per iteration over the classical Q-learning algorithm.

The contraction mapping analysis (Prop. 6.6.1) for SSP problems in Section 6.6 is based on the convergence analysis for TD(λ) given in Bertsekas and Tsitsiklis [BeT96], Section 6.3.4. The LSPE algorithm was first proposed for SSP problems in [BeI96] as an implementation of the λ-policy iteration method of Section 6.3.9 (see also [Ber11b]).

The TD(λ) algorithm was extended to the average cost problem, and its convergence was proved by Tsitsiklis and Van Roy [TsV99a] (see also [TsV02]). The average cost analysis of LSPE in Section 6.7.1 is due to Yu and Bertsekas [YuB06b]. An alternative to the LSPE and LSTD algorithms of Section 6.7.1 is based on the relation between average cost and SSP problems, and the associated contracting value iteration method discussed in Section 4.4.1. The idea is to convert the average cost problem into a parametric form of SSP, which however converges to the correct one as the gain of the policy is estimated correctly by simulation. The SSP algorithms of Section 6.6 can then be used with the estimated gain of the policy η_k replacing the true gain η.

While the convergence analysis of the policy evaluation methods of Sections 6.3 and 6.6 is based on contraction mapping arguments, a different line of analysis is necessary for Q-learning algorithms for average cost problems (as well as for SSP problems where there may exist some improper policies). The reason is that there may not be an underlying contraction, so the nonexpansive property of the DP mapping must be used instead. As a result, the analysis is more complicated, and a different method of proof has been employed, based on the so-called ODE approach; see Abounadi, Bertsekas, and Borkar [ABB01], [ABB02], and Borkar and Meyn [BeM00]. In particular, the Q-learning algorithms of Section 6.7.3 were proposed and analyzed in these references. They are also discussed in the book [BeT96] (Section 7.1.5). Alternative algorithms of the Q-learning type for average cost problems were given without convergence proof by Schwartz [Sch93b], Singh [Sin94], and Mahadevan [Mah96]; see also Gosavi [Gos04].

The framework of Sections 6.8.1-6.8.6 on generalized projected equation and Bellman error methods is based on Bertsekas and Yu [BeY07], [BeY09], which also discuss in greater detail multistep methods, and several other variants of the methods given here (see also Bertsekas [Ber09b]). The regression-based method and the confidence interval analysis of Prop. 6.8.1 is due to Wang, Polydorides, and Bertsekas [WPB09]. The material of Section 6.8.7 on oblique projections and the connections to aggregation/discretization with a coarse grid is based on unpublished collaboration with H. Yu. The generalized aggregation methodology of Section 6.8.8 is new in the form given here, but is motivated by the development of aggregation-based approximate DP given in Section 6.4.

The paper by Yu and Bertsekas [YuB08] derives error bounds which apply to generalized projected equations and sharpen the rather conservative bound

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi, \tag{6.297}$$

given for discounted DP problems (cf. Prop. 6.3.2) and the bound

$$\|x^* - \Phi r^*\| \leq \left\|(I - \Pi A)^{-1}\right\| \|x^* - \Pi x^*\|,$$

for the general projected equation $\Phi r = \Pi(A\Phi r + b)$ [cf. Eq. (6.242)]. The bounds of [YuB08] apply also to the case where A is not a contraction and have the form

$$\|x^* - \Phi r^*\|_\xi \leq B(A, \xi, S) \|x^* - \Pi x^*\|_\xi,$$

where $B(A, \xi, S)$ is a scalar that [contrary to the scalar $1/\sqrt{1-\alpha^2}$ in Eq. (6.297)] depends on the approximation subspace S and the structure of the matrix A. The scalar $B(A, \xi, S)$ involves the spectral radii of some low-dimensional matrices and may be computed either analytically or by simulation (in the case where x has large dimension). One of the scalars $B(A, \xi, S)$ given in [YuB08] involves only the matrices that are computed as part of the simulation-based calculation of the matrix C_k via Eq. (6.249), so it is simply obtained as a byproduct of the LSTD and LSPE-type methods of Section 6.8.1. Among other situations, such bounds can be useful in cases where the "bias" $\|\Phi r^* - \Pi x^*\|_\xi$ (the distance between the solution Φr^* of the projected equation and the best approximation of x^* within S, which is Πx^*) is very large [cf., the example of Exercise 6.9, mentioned earlier, where TD(0) produces a very bad solution relative to TD(λ) for $\lambda \approx 1$]. A value of $B(A, \xi, S)$ that is much larger than 1 strongly suggests a large bias and motivates corrective measures (e.g., increase λ in the approximate DP case, changing the subspace S, or changing ξ). Such an inference cannot be made based on the much less discriminating bound (6.297), even if A is a contraction with respect to $\|\cdot\|_\xi$.

The Bellman equation error approach was initially suggested by Schweitzer and Seidman [ScS85], and simulation-based algorithms based on this approach were given later by Harmon, Baird, and Klopf [HBK94], Baird [Bai95], and Bertsekas [Ber95], including the two-sample simulation-based method for policy evaluation based on minimization of the Bellman equation error (Section 6.8.5 and Fig. 6.8.2). For some recent developments, see Ormoneit and Sen [OrS02], Szepesvari and Smart [SzS04], Antos, Szepesvari, and Munos [ASM08], Bethke, How, and Ozdaglar [BHO08], and Scherrer [Sch10].

There is a large literature on policy gradient methods for average cost problems. The formula for the gradient of the average cost has been given in different forms and within a variety of different contexts: see Cao and Chen [CaC97], Cao and Wan [CaW98], Cao [Cao99], [Cao05], Fu and Hu [FuH94], Glynn [Gly87], Jaakkola, Singh, and Jordan [JSJ95], L'Ecuyer [L'Ec91], and Williams [Wil92]. We follow the derivations of Marbach and Tsitsiklis [MaT01]. The inner product expression of $\partial\eta(r)/\partial r_m$ was used to delineate essential features for gradient calculation by Konda and Tsitsiklis [KoT99], [KoT03], and Sutton, McAllester, Singh, and Mansour [SMS99].

Several implementations of policy gradient methods, some of which use cost approximations, have been proposed: see Cao [Cao04], Grudic and Ungar [GrU04], He [He02], He, Fu, and Marcus [HFM05], Kakade [Kak02], Konda [Kon02], Konda and Borkar [KoB99], Konda and Tsitsiklis [KoT99], [KoT03], Marbach and Tsitsiklis [MaT01], [MaT03], Sutton, McAllester, Singh, and Mansour [SMS99], and Williams [Wil92].

Approximation in policy space can also be carried out very simply by a random search method in the space of policy parameters. There has been considerable progress in random search methodology, and the cross-entropy method (see Rubinstein and Kroese [RuK04], [RuK08], de Boer et al [BKM05]) has gained considerable attention. A noteworthy success with this method has been attained in learning a high scoring strategy in the game of tetris (see Szita and Lorinz [SzL06], and Thiery and Scherrer [ThS09]); surprisingly this method outperformed in terms of scoring performance methods based on approximate policy iteration, approximate linear programming, and policy gradient by more than an order of magnitude (see the discussion of policy oscillations and chattering in Section 6.3.8). Other random search algorithms have also been suggested; see Chang, Fu, Hu, and Marcus [CFH07], Ch. 3. Additionally, statistical inference methods have been adapted for approximation in policy space in the context of some special applications, with the policy parameters viewed as the parameters in a corresponding inference problem; see Attias [Att03], Toussaint and Storey [ToS06], and Verma and Rao [VeR06].

Approximate DP methods for partially observed Markov decision problems (POMDP) are not as well-developed as their perfect observation counterparts. Approximations obtained by aggregation/interpolation schemes and solution of finite-spaces discounted or average cost problems

have been proposed by Zhang and Liu [ZhL97], Zhou and Hansen [ZhH01], and Yu and Bertsekas [YuB04] (see Example 6.4.1); see also Zhou, Fu, and Marcus [ZFM10]. Alternative approximation schemes based on finite-state controllers are analyzed in Hauskrecht [Hau00], Poupart and Boutilier [PoB04], and Yu and Bertsekas [YuB06a]. Policy gradient methods of the actor-only type have been given by Baxter and Bartlett [BaB01], and Aberdeen and Baxter [AbB00]. An alternative method, which is of the actor-critic type, has been proposed by Yu [Yu05]. See also Singh, Jaakkola, and Jordan [SJJ94], and Moazzez-Estanjini, Li, and Paschalidis [ELP09].

Many problems have special structure, which can be exploited in approximate DP. For some representative work, see Guestrin et al. [GKP03], and Koller, and Parr [KoP00].

EXERCISES

6.1

Consider a fully connected network with n nodes, and the problem of finding a travel strategy that takes a traveller from node 1 to node n in no more than a given number m of time periods, while minimizing the expected travel cost (sum of the travel costs of the arcs on the travel path). The cost of traversing an arc changes randomly and independently at each time period with given distribution. For any node i, the current cost of traversing the outgoing arcs (i, j), $j \neq i$, will become known to the traveller upon reaching i, who will then either choose the next node j on the travel path, or stay at i (waiting for smaller costs of outgoing arcs at the next time period) at a fixed (deterministic) cost per period. Derive a DP algorithm in a space of post-decision variables and compare it to ordinary DP.

6.2 (Multiple State Visits in Monte Carlo Simulation)

Argue that the Monte Carlo simulation formula

$$J_\mu(i) = \lim_{M\to\infty} \frac{1}{M} \sum_{m=1}^{M} c(i,m)$$

is valid even if a state may be revisited within the same sample trajectory. *Note*: If only a finite number of trajectories is generated, in which case the number M of cost samples collected for a given state i is finite *and* random, the sum $\frac{1}{M}\sum_{m=1}^{M} c(i,m)$ need not be an unbiased estimator of $J_\mu(i)$. However, as the number of trajectories increases to infinity, the bias disappears. See [BeT96], Sections 5.1, 5.2, for a discussion and examples. *Hint*: Suppose the M cost samples are generated from N trajectories, and that the kth trajectory involves n_k visits to state i and generates n_k corresponding cost samples. Denote $m_k = n_1 + \cdots + n_k$. Write

$$\lim_{M\to\infty} \frac{1}{M} \sum_{m=1}^{M} c(i,m) = \lim_{N\to\infty} \frac{\frac{1}{N}\sum_{k=1}^{N} \sum_{m=m_{k-1}+1}^{m_k} c(i,m)}{\frac{1}{N}(n_1 + \cdots + n_N)}$$

$$= \frac{E\left\{\sum_{m=m_{k-1}+1}^{m_k} c(i,m)\right\}}{E\{n_k\}},$$

and argue that

$$E\left\{\sum_{m=m_{k-1}+1}^{m_k} c(i,m)\right\} = E\{n_k\} J_\mu(i),$$

(or see Ross [Ros83b], Cor. 7.2.3 for a closely related result).

6.3 (Viewing Q-Factors as Optimal Costs)

Consider the stochastic shortest path problem under Assumptions 2.1.1 and 2.1.2. Show that the Q-factors $Q(i,u)$ can be viewed as state costs associated with a modified stochastic shortest path problem. Use this fact to show that the Q-factors $Q(i,u)$ are the unique solution of the system of equations

$$Q(i,u) = \sum_j p_{ij}(u) \left(g(i,u,j) + \min_{v \in U(j)} Q(j,v) \right).$$

Hint: Introduce a new state for each pair (i,u), with transition probabilities $p_{ij}(u)$ to the states $j = 1, \ldots, n, t$.

6.4

This exercise provides a counterexample to the convergence of PVI for discounted problems when the projection is with respect to a norm other than $\|\cdot\|_\xi$. Consider the mapping $TJ = g + \alpha PJ$ and the algorithm $\Phi r_{k+1} = \Pi T(\Phi r_k)$, where P and Φ satisfy Assumptions 6.3.1 and 6.3.2. Here Π denotes projection on the subspace spanned by Φ with respect to the weighted Euclidean norm $\|J\|_v = \sqrt{J'VJ}$, where V is a diagonal matrix with positive components. Use the formula $\Pi = \Phi(\Phi' V \Phi)^{-1} \Phi' V$ to show that in the single basis function case (Φ is an $n \times 1$ vector) the algorithm is written as

$$r_{k+1} = \frac{\Phi' V g}{\Phi' V \Phi} + \frac{\alpha \Phi' V P \Phi}{\Phi' V \Phi} r_k.$$

Construct choices of α, g, P, Φ, and V for which the algorithm diverges.

6.5 (LSPE(0) for Average Cost Problems [YuB06b])

Show the convergence of LSPE(0) for average cost problems with unit stepsize, assuming that P is aperiodic, by showing that the eigenvalues of the matrix ΠF lie strictly within the unit circle.

6.6 (Relation of Discounted and Average Cost Approximations [TsV02])

Consider the finite-state α-discounted and average cost frameworks of Sections 6.3 and 6.7 for a fixed stationary policy with cost per stage g and transition probability matrix P. Assume that the states form a single recurrent class, let J_α be the α-discounted cost vector, let (η^*, h^*) be the gain-bias pair, let ξ be the steady-state probability vector, let Ξ be the diagonal matrix with diagonal elements the components of ξ, and let

$$P^* = \lim_{N \to \infty} \sum_{k=0}^{N-1} P^k.$$

Show that:

(a) $\eta^* = (1-\alpha)\xi' J_\alpha$ and $P^* J_\alpha = (1-\alpha)^{-1}\eta^* e$.

(b) $h^* = \lim_{\alpha\to 1}(I - P^*)J_\alpha$. *Hint*: Use the Laurent series expansion of J_α (cf. Prop. 4.1.2).

(c) Consider the subspace

$$E^* = \big\{(I - P^*)y \mid y \in \Re^n\big\},$$

which is orthogonal to the unit vector e in the scaled geometry where x and y are orthogonal if $x'\Xi y = 0$ (cf. Fig. 6.7.1). Verify that J_α can be decomposed into the sum of two vectors that are orthogonal (in the scaled geometry): $P^* J_\alpha$, which is the projection of J_α onto the line defined by e, and $(I - P^*)J_\alpha$, which is the projection of J_α onto E^* and converges to h^* as $\alpha \to 1$.

(d) Use part (c) to show that the limit $r^*_{\lambda,\alpha}$ of PVI(λ) for the α-discounted problem converges to the limit r^*_λ of PVI(λ) for the average cost problem as $\alpha \to 1$.

6.7 (Conversion of SSP to Average Cost Policy Evaluation)

We have often used the transformation of an average cost problem to an SSP problem (cf. Section 4.3.1, and Chapter 7 of Vol. I). The purpose of this exercise (unpublished collaboration of H. Yu and the author) is to show that a reverse transformation is possible, from SSP to average cost, at least in the case where all policies are proper. As a result, analysis, insights, and algorithms for average cost policy evaluation can be applied to policy evaluation of a SSP problem.

Consider the SSP problem, a single proper stationary policy μ, and the probability distribution $q_0 = \big(q_0(1), \ldots, q_0(n)\big)$ used for restarting simulated trajectories [cf. Eq. (6.216)]. Let us modify the Markov chain by eliminating the self-transition from state 0 to itself, and substituting instead transitions from 0 to i with probabilities $q_0(i)$,

$$\tilde{p}_{0i} = q_0(i),$$

each with a fixed transition cost β, where β is a scalar parameter. All other transitions and costs remain the same (cf. Fig. 6.10.1). We call the corresponding average cost problem β-AC. Denote by J_μ the SSP cost vector of μ, and by η_β and $h_\beta(i)$ the average and differential costs of β-AC, respectively.

(a) Show that η_β can be expressed as the average cost per stage of the cycle that starts at state 0 and returns to 0, i.e.,

$$\eta_\beta = \frac{\beta + \sum_{i=1}^n q_0(i) J_\mu(i)}{T},$$

where T is the expected time to return to 0 starting from 0.

(b) Show that for the special value

$$\beta^* = -\sum_{i=1}^n q_0(i) J_\mu(i),$$

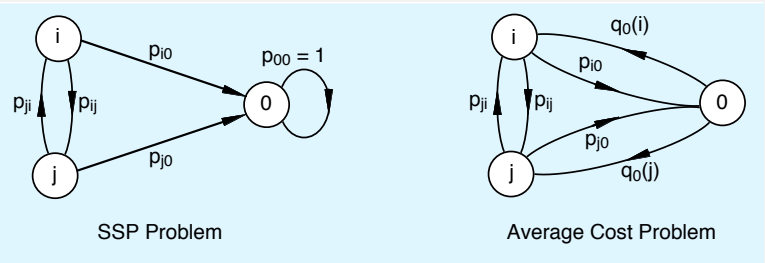

Figure 6.10.1 Transformation of a SSP problem to an average cost problem. The transitions from 0 to each $i = 1, \ldots, n$, have cost β.

we have $\eta_{\beta^*} = 0$, and

$$J_\mu(i) = h_{\beta^*}(i) - h_{\beta^*}(0), \qquad i = 1, \ldots, n.$$

Hint: Since the states of β-AC form a single recurrent class, we have from Bellman's equation

$$\eta_\beta + h_\beta(i) = \sum_{j=0}^{n} p_{ij}\big(g(i,j) + h_\beta(j)\big), \qquad i = 1, \ldots, n, \tag{6.298}$$

$$\eta_\beta + h_\beta(0) = \beta + \sum_{i=1}^{n} q_0(i) h_\beta(i). \tag{6.299}$$

From Eq. (6.298) it follows that if $\beta = \beta^*$, we have $\eta_{\beta^*} = 0$, and

$$\delta(i) = \sum_{j=0}^{n} p_{ij} g(i,j) + \sum_{j=1}^{n} p_{ij}\delta(j), \qquad i = 1, \ldots, n, \tag{6.300}$$

where

$$\delta(i) = h_{\beta^*}(i) - h_{\beta^*}(0), \qquad i = 1, \ldots, n.$$

Since Eq. (6.300) is Bellman's equation for the SSP problem, we see that $\delta(i) = J_\mu(i)$ for all i.

(c) Derive a transformation to convert an average cost policy evaluation problem into another average cost policy evaluation problem where the transition probabilities out of a single state are modified in any way such that the states of the resulting Markov chain form a single recurrent class. The two average cost problems should have the same differential cost vectors, except for a constant shift. *Note*: This conversion may be useful if the transformed problem has more favorable properties.

6.8 (Projected Equations for Finite-Horizon Problems)

Consider a finite-state finite-horizon policy evaluation problem with the cost vector and transition matrices at time m denoted by g_m and P_m, respectively. The DP algorithm/Bellman's equation takes the form

$$J_m = g_m + P_m J_{m+1}, \qquad m = 0, \ldots, N-1,$$

where J_m is the cost vector of stage m for the given policy, and J_N is a given terminal cost vector. Consider a low-dimensional approximation of J_m that has the form

$$J_m \approx \Phi_m r_m, \qquad m = 0, \ldots, N-1,$$

where Φ_m is a matrix whose columns are basis functions. Consider also a projected equation of the form

$$\Phi_m r_m = \Pi_m(g_m + P_m \Phi_{m+1} r_{m+1}), \qquad m = 0, \ldots, N-1,$$

where Π_m denotes projection onto the space spanned by the columns of Φ_m with respect to a weighted Euclidean norm with weight vector ξ_m.

(a) Show that the projected equation can be written in the equivalent form

$$\Phi_m' \Xi_m(\Phi_m r_m - g_m - P_m \Phi_{m+1} r_{m+1}) = 0, \qquad m = 0, \ldots, N-2,$$

$$\Phi_{N-1}' \Xi_{N-1}(\Phi_{N-1} r_{N-1} - g_{N-1} - P_{N-1} J_N) = 0,$$

where Ξ_m is the diagonal matrix having the vector ξ_m along the diagonal. *Abbreviated solution*: The derivation follows the one of Section 6.3.1 [cf. the analysis leading to Eqs. (6.40) and (6.41)]. The solution $\{r_0^*, \ldots, r_{N-1}^*\}$ of the projected equation is obtained as

$$r_m^* = \arg \min_{r_0, \ldots, r_{N-1}} \left\{ \sum_{m=0}^{N-2} \left\| \Phi_m r_m - (g_m + P_m \Phi_{m+1} r_{m+1}^*) \right\|_{\xi_m}^2 + \left\| \Phi_{N-1} r_{N-1} - (g_{N-1} + P_{N-1} J_N) \right\|_{\xi_{N-1}}^2 \right\}.$$

The minimization can be decomposed into N minimizations, one for each m, and by setting to 0 the gradient with respect to r_m, we obtain the desired form.

(b) Consider a simulation scheme that generates a sequence of trajectories of the system, similar to the case of a stochastic shortest path problem (cf. Section 6.6). Derive corresponding LSTD and (scaled) LSPE algorithms.

(c) Derive appropriate modifications of the algorithms of Section 6.5.3 to address a finite horizon version of the optimal stopping problem.

6.9 (Approximation Error of TD Methods [Ber95])

This exercise illustrates how the value of λ may significantly affect the approximation quality in TD methods. Consider a problem of the SSP type, but with a single policy. The states are $0, 1, \ldots, n$, with state 0 being the termination state. Under the given policy, the system moves deterministically from state $i \geq 1$ to state $i-1$ at a cost g_i. Consider a linear approximation of the form

$$\tilde{J}(i, r) = i\, r$$

for the cost-to-go function, and the application of TD methods. Let all simulation runs start at state n and end at 0 after visiting all the states $n-1, n-2, \ldots, 1$ in succession.

(a) Derive the corresponding projected equation $\Phi r_\lambda^* = \Pi T^{(\lambda)}(\Phi r_\lambda^*)$ and show that its unique solution r_λ^* satisfies

$$\sum_{k=1}^{n} (g_k - r_\lambda^*)\big(\lambda^{n-k} n + \lambda^{n-k-1}(n-1) + \cdots + k\big) = 0.$$

(b) Plot $\tilde{J}(i, r_\lambda^*)$ with λ from 0 to 1 in increments of 0.2, for the following two cases:

(1) $n = 50$, $g_1 = 1$ and $g_i = 0$ for all $i \neq 1$.

(2) $n = 50$, $g_n = -(n-1)$ and $g_i = 1$ for all $i \neq n$.

Figure 6.10.2 gives the results for $\lambda = 0$ and $\lambda = 1$.

6.11

This exercise provides an example of comparison of the projected equation approach of Section 6.8.1 and the least squares approach of Section 6.8.2. Consider the case of a linear system involving a vector x with two block components, $x_1 \in \Re^k$ and $x_2 \in \Re^m$. The system has the form

$$x_1 = A_{11} x_1 + b_1, \qquad x_2 = A_{21} x_1 + A_{22} x_2 + b_2,$$

so x_1 can be obtained by solving the first equation. Let the approximation subspace be $\Re^k \times S_2$, where S_2 is a subspace of $\Re^m$. Show that with the projected equation approach, we obtain the component x_1^* of the solution of the original equation, but with the least squares approach, we do not.

6.12 (Error Bounds for Hard Aggregation [TsV96])

Consider the hard aggregation case of Section 6.4.2, and denote $i \in x$ if the original state i belongs to aggregate state x. Also for every i denote by $x(i)$ the aggregate state x with $i \in x$. Consider the corresponding mapping F defined by

$$(FR)(x) = \sum_{i=1}^{n} d_{xi} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i, u, j) + \alpha R\big(x(j)\big)\Big), \qquad x \in A,$$

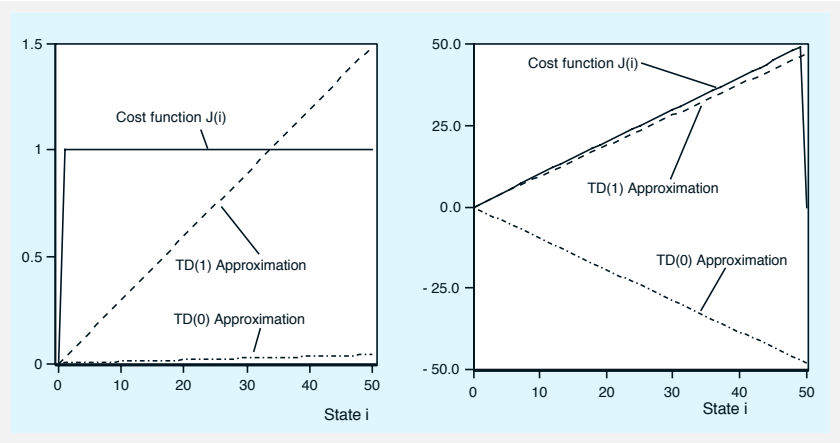

Figure 6.10.2 Form of the cost-to-go function $J(i)$, and the linear representations $\tilde{J}(i, r^*_\lambda)$ in Exercise 6.9, for the case

$$g_1 = 1, \qquad g_i = 0, \quad \forall\, i \neq 1$$

(figure on the left), and the case.

$$g_n = -(n-1), \qquad g_i = 1, \quad \forall\, i \neq n$$

(figure on the right).

[cf. Eq. (6.162)], and let R^* be the unique fixed point of this mapping. Show that

$$R^*(x) - \frac{\epsilon}{1-\alpha} \leq J^*(i) \leq R^*(x) + \frac{\epsilon}{1-\alpha}, \qquad \forall\, x \in A,\ i \in x,$$

where

$$\epsilon = \max_{x \in A} \max_{i,j \in x} \big|J^*(i) - J^*(j)\big|.$$

Abbreviated Proof: Let the vector $\overline{R}$ be defined by

$$\overline{R}(x) = \min_{i \in x} J^*(i) + \frac{\epsilon}{1-\alpha}, \qquad x \in A.$$

We have for all $x \in A$,

$$\begin{aligned}
(F\overline{R})(x) &= \sum_{i=1}^{n} d_{xi} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha \overline{R}\big(x(j)\big)\Big) \\
&\leq \sum_{i=1}^{n} d_{xi} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha J^*(j) + \frac{\alpha\epsilon}{1-\alpha}\right) \\
&= \sum_{i=1}^{n} d_{xi}\left(J^*(i) + \frac{\alpha\epsilon}{1-\alpha}\right) \\
&\leq \min_{i \in x}\big(J^*(i) + \epsilon\big) + \frac{\alpha\epsilon}{1-\alpha} \\
&= \min_{i \in x} J^*(i) + \frac{\epsilon}{1-\alpha} \\
&= \overline{R}(x).
\end{aligned}$$

Thus, $F\overline{R} \leq \overline{R}$, from which it follows that $R^* \leq \overline{R}$ (since $R^* = \lim_{k\to\infty} F^k\overline{R}$ and F is monotone). This proves the left-hand side of the desired inequality. The right-hand side follows similarly.

6.13 (Hard Aggregation as a Projected Equation Method)

Consider a fixed point equation of the form

$$r = DT(\Phi r),$$

where $T : \Re^n \mapsto \Re^n$ is a (possibly nonlinear) mapping, and D and Φ are $s \times n$ and $n \times s$ matrices, respectively, and Φ has rank s. Writing this equation as

$$\Phi r = \Phi DT(\Phi r),$$

we see that it is a projected equation if ΦD is a projection onto the subspace $S = \{\Phi r \mid r \in \Re^s\}$ with respect to a weighted Euclidean norm. The purpose of this exercise is to prove that this is the case in hard aggregation schemes, where the set of indices $\{1, \ldots, n\}$ is partitioned into s disjoint subsets $I_1, \ldots, I_s$ and:

(1) The ℓth column of Φ has components that are 1 or 0 depending on whether they correspond to an index in I_ℓ or not.

(2) The ℓth row of D is a probability distribution $(d_{\ell 1}, \ldots, d_{\ell n})$ whose components are positive depending on whether they correspond to an index in I_ℓ or not, i.e., $\sum_{i=1}^{n} d_{\ell i} = 1$, $d_{\ell i} > 0$ if $i \in I_\ell$, and $d_{\ell i} = 0$ if $i \notin I_\ell$.

Show in particular that ΦD is given by the projection formula

$$\Phi D = \Phi(\Phi' \Xi \Phi)^{-1} \Phi' \Xi,$$

where Ξ is the diagonal matrix with the nonzero components of D along the diagonal, normalized so that they form a probability distribution, i.e.,

$$\xi_i = \frac{d_{\ell i}}{\sum_{k=1}^{s} \sum_{j=1}^{n} d_{kj}}, \qquad \forall\ i \in I_\ell,\ \ell = 1, \ldots, s.$$

Notes: (1) Based on the preceding result, if T is a contraction mapping with respect to the projection norm, the same is true for ΦDT. In addition, if T is a contraction mapping with respect to the sup-norm, the same is true for $DT\Phi$ (since aggregation and disaggregation matrices are nonexpansive with respect to the sup-norm); this is true for all aggregation schemes, not just hard aggregation. (2) For ΦD to be a weighted Euclidean projection, we must have $\Phi D\Phi D = \Phi D$. This implies that if $D\Phi$ is invertible and ΦD is a weighted Euclidean projection, we must have $D\Phi = I$ (since if $D\Phi$ is invertible, Φ has rank s, which implies that $D\Phi D = D$ and hence $D\Phi = I$, since D also has rank s). From this it can be seen that out of all possible aggregation schemes with $D\Phi$ invertible and D having nonzero columns, only hard aggregation has the projection property of this exercise.

6.14 (Simulation-Based Implementation of Linear Aggregation Schemes)

Consider a linear system $Ax = b$, where A is an $n \times n$ matrix and b is a column vector in $\Re^n$. In a scheme that generalizes the aggregation approach of Section 6.5, we introduce an $n \times s$ matrix Φ, whose columns are viewed as basis functions, and an $s \times n$ matrix D. We find a solution r^* of the $s \times s$ system

$$DA\Phi r = Db,$$

and we view Φr^* as an approximate solution of $Ax = b$. An approximate implementation is to compute by simulation approximations $\tilde{C}$ and $\tilde{d}$ of the matrices $C = DA\Phi$ and $d = Db$, respectively, and then solve the system $\tilde{C}r = \tilde{d}$. The purpose of the exercise is to provide a scheme for doing this.

Let $\overline{D}$ and $\overline{A}$ be matrices of dimensions $s \times n$ and $n \times n$, respectively, whose rows are probability distributions, and are such that their components satisfy

$$\overline{G}_{\ell i} > 0 \qquad \text{if } G_{\ell i} \neq 0, \qquad \ell = 1, \ldots, s, \ i = 1, \ldots, n,$$

$$\overline{A}_{ij} > 0 \qquad \text{if } A_{ij} \neq 0, \qquad i = 1, \ldots, n, \ j = 1, \ldots, n.$$

We approximate the (ℓm)th component $C_{\ell m}$ of C as follows. We generate a sequence $\big\{(i_t, j_t) \mid t = 1, 2, \ldots\big\}$ by independently generating each i_t according to the distribution $\{\overline{G}_{\ell i} \mid i = 1, \ldots, n\}$, and then by independently generating j_t according to the distribution $\{\overline{A}_{i_t j} \mid j = 1, \ldots, n\}$. For $k = 1, 2, \ldots$, consider the scalar

$$C^k_{\ell m} = \frac{1}{k} \sum_{t=1}^{k} \frac{G_{\ell i_t}}{\overline{G}_{\ell i_t}} \frac{A_{i_t j_t}}{\overline{A}_{i_t j_t}} \Phi_{j_t m},$$

where Φ_{jm} denotes the (jm)th component of Φ. Show that with probability 1 we have

$$\lim_{k \to \infty} C^k_{\ell m} = C_{\ell m}.$$

Derive a similar scheme for approximating the components of d.

6.15 (Approximate Policy Iteration Using an Approximate Problem)

Consider the discounted problem of Section 6.3 (referred to as DP) and an approximation to this problem (this is a different discounted problem referred to as AP). This exercise considers an approximate policy iteration method where the policy evaluation is done through AP, but the policy improvement is done through DP – a process that is patterned after the aggregation-based policy iteration method of Section 6.4.2. In particular, we assume that the two problems, DP and AP, are connected as follows:

(1) DP and AP have the same state and control spaces, and the same policies.

(2) For any policy μ, its cost vector in AP, denoted $\tilde{J}_\mu$, satisfies

$$\|\tilde{J}_\mu - J_\mu\|_\infty \leq \delta,$$

i.e., policy evaluation using AP, rather than DP, incurs an error of at most δ in sup-norm.

(3) The policy $\overline{\mu}$ obtained by exact policy iteration in AP satisfies the equation

$$T\tilde{J}_{\overline{\mu}} = T_{\overline{\mu}}\tilde{J}_{\overline{\mu}}.$$

This is true in particular if the policy improvement process in AP is identical to the one in DP.

Show the error bound

$$\|J_{\overline{\mu}} - J^*\|_\infty \leq \frac{2\alpha\delta}{1-\alpha}e$$

[cf. Eq. (6.135)]. *Hint*: Follow the derivation of the error bound (6.135).

6.16 (Approximate Policy Iteration and Newton's Method)

Consider the discounted problem, and a policy iteration method that uses function approximation over a subspace $S = \{\Phi r \mid r \in \Re^s\}$, and operates as follows: Given a vector $r_k \in \Re^s$, define $\overline{\mu}_k$ to be a policy such that

$$T_{\overline{\mu}_k}(\Phi r_k) = T(\Phi r_k), \tag{6.301}$$

and let r_{k+1} satisfy

$$r_{k+1} = L_{k+1}T_{\overline{\mu}_k}(\Phi r_{k+1}),$$

where L_{k+1} is an $s \times n$ matrix. Show that if $\overline{\mu}_k$ is the unique policy satisfying Eq. (6.301), then r_{k+1} is obtained from r_k by a Newton step for solving the equation

$$r = L_{k+1}T(\Phi r),$$

(cf. Exercise 1.10).

6.17 (Projected Equations for Approximation of Cost Function Differences)

Let x^* be the unique solution of an equation of the form $x = b + Ax$. Suppose that we are interested in approximating within a subspace $S = \{\Phi r \mid r \in \Re^s\}$ the vector $y^* = Dx^*$, where D is an invertible matrix.

(a) Show that y^* is the unique solution of $y = c + By$, where

$$B = DAD^{-1}, \qquad c = Db.$$

(b) Consider the projected equation approach of approximating y^* by Φr^*, obtained by solving $\Phi r = \Pi(c + B\Phi r)$ (cf. Section 6.8.1), and the special case where D is the matrix that maps $\big(x(1), \ldots, x(n)\big)$ to the vector $\big(y(1), \ldots, y(n)\big)$, with y consisting of component differences of x: $y(i) = x(i) - x(n)$, $i = 1, \ldots, n-1$, and $y(n) = x(n)$. Calculate D, B, and c, and develop a simulation-based matrix inversion approach for this case.

6.18 (Constrained Projected Equations [Ber09b], [Ber11a])

Consider the projected equation $J = \Pi TJ$, where the projection Π is done on a closed convex subset $\hat{S}$ of the approximation subspace $S = \{\Phi r \mid r \in \Re^s\}$ (rather than on S itself).

(a) Show that the projected equation is equivalent to finding $r^* \in \Re^s$ such that

$$f(\Phi r^*)'\Phi(r - r^*) \geq 0, \qquad \forall\ r \in \hat{R},$$

where

$$f(J) = \Xi(J - TJ), \qquad \hat{R} = \{r \mid \Phi r \in \hat{S}\}.$$

Note: This type of inequality is known as a *variational inequality*.

(b) Consider the special case where Φ is partitioned as $[\phi_1\ \hat{\Phi}]$ where ϕ_1 is the first column of Φ and $\hat{\Phi}$ is the $n \times (s-1)$ matrix comprising the remaining $(s-1)$ columns. Let $\hat{S}$ be the affine subset of S given by

$$\hat{S} = \{\beta + \hat{\Phi}\hat{r} \mid \hat{r} \in \Re^{s-1}\},$$

where β is a fixed multiple of the vector ϕ_1. Let $TJ = g + AJ$ where A is an $n \times n$ matrix. Show that the projected equation is equivalent to finding $\hat{r} \in \Re^{s-1}$ that solves the equation $\hat{C}\hat{r} = \hat{d}$, where

$$\hat{C} = \hat{\Phi}'\Xi(I - A)\hat{\Phi}, \qquad \hat{d} = \hat{\Phi}'\Xi(g + A\phi_1 - \phi_1).$$

Derive an LSTD-type algorithm to obtain simulation-based approximations to $\hat{C}$ and $\hat{d}$, and corresponding approximation to $\hat{r}$.

6.19 (Policy Gradient Formulas for SSP)

Consider the SSP context, and let the cost per stage and transition probability matrix be given as functions of a parameter vector r. Denote by $g_i(r)$, $i = 1, \ldots, n$, the expected cost starting at state i, and by $p_{ij}(r)$ the transition probabilities. Each value of r defines a stationary policy, which is assumed proper. For each r, the expected costs starting at states i are denoted by $J_i(r)$. We wish to calculate the gradient of a weighted sum of the costs $J_i(r)$, i.e.,

$$\bar{J}(r) = \sum_{i=1}^{n} q(i) J_i(r),$$

where $q = \big(q(1), \ldots, q(n)\big)$ is some probability distribution over the states. Consider a single scalar component r_m of r, and differentiate Bellman's equation to show that

$$\frac{\partial J_i}{\partial r_m} = \frac{\partial g_i}{\partial r_m} + \sum_{j=1}^{n} \frac{\partial p_{ij}}{\partial r_m} J_j + \sum_{j=1}^{n} p_{ij} \frac{\partial J_j}{\partial r_m}, \qquad i = 1, \ldots, n,$$

where the argument r at which the partial derivatives are computed is suppressed. Interpret the above equation as Bellman's equation for a SSP problem.

References

[ABB01] Abounadi, J., Bertsekas, B. P., and Borkar, V. S., 2001. "Learning Algorithms for Markov Decision Processes with Average Cost," SIAM J. on Control and Optimization, Vol. 40, pp. 681-698.

[ABB02] Abounadi, J., Bertsekas, B. P., and Borkar, V. S., 2002. "Stochastic Approximation for Non-Expansive Maps: Q-Learning Algorithms," SIAM J. on Control and Optimization, Vol. 41, pp. 1-22.

[ABJ06] Ahamed, T. P. I., Borkar, V. S., and Juneja, S., 2006. "Adaptive Importance Sampling Technique for Markov Chains Using Stochastic Approximation," Operations Research, Vol. 54, pp. 489-504.

[ASM08] Antos, A., Szepesvari, C., and Munos, R., 2008. "Learning Near-Optimal Policies with Bellman-Residual Minimization Based Fitted Policy Iteration and a Single Sample Path," Vol. 71, pp. 89-129.

[AbB02] Aberdeen, D., and Baxter, J., 2002. "Scalable Internal-State Policy-Gradient Methods for POMDPs," Proc. of the Nineteenth International Conference on Machine Learning, pp. 3-10.

[Ama98] Amari, S., 1998. "Natural Gradient Works Efficiently in Learning," Neural Computation, Vol. 10, pp. 251-276.

[Att03] Attias, H. 2003. "Planning by Probabilistic Inference," in C. M. Bishop and B. J. Frey, (Eds.), Proc. of the 9th Int. Workshop on Artificial Intelligence and Statistics.

[BBN04] Bertsekas, D. P., Borkar, V., 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, (Eds.), IEEE Press, N. Y.

[BBS95] Barto, A. G., Bradtke, S. J., and Singh, S. P., 1995. "Real-Time Learning and Control Using Asynchronous Dynamic Programming," Artificial Intelligence, Vol. 72, pp. 81-138.

[BBD10] Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D., 2010. Reinforcement Learning and Dynamic Programming Using Function Ap-

proximators, CRC Press, N. Y.

[BED09] Busoniu, L., Ernst, D., De Schutter, B., and Babuska, R., 2009. "Online Least-Squares Policy Iteration for Reinforcement Learning Control," unpublished report, Delft Univ. of Technology, Delft, NL.

[BHO08] Bethke, B., How, J. P., and Ozdaglar, A., 2008. "Approximate Dynamic Programming Using Support Vector Regression," Proc. IEEE Conference on Decision and Control, Cancun, Mexico.

[BKM05] de Boer, P. T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. 2005. "A Tutorial on the Cross-Entropy Method," Annals of Operations Research, Vol. 134, pp. 19-67.

[BMP90] Benveniste, A., Metivier, M., and Priouret, P., 1990. Adaptive Algorithms and Stochastic Approximations, Springer-Verlag, N. Y.

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

[BaB01] Baxter, J., and Bartlett, P. L., 2001. "Infinite-Horizon Policy-Gradient Estimation," J. Artificial Intelligence Research, Vol. 15, pp. 319–350.

[Bai93] Baird, L. C., 1993. "Advantage Updating," Report WL-TR-93-1146, Wright Patterson AFB, OH.

[Bai94] Baird, L. C., 1994. "Reinforcement Learning in Continuous Time: Advantage Updating," International Conf. on Neural Networks, Orlando, Fla.

[Bai95] Baird, L. C., 1995. "Residual Algorithms: Reinforcement Learning with Function Approximation," Dept. of Computer Science Report, U.S. Air Force Academy, CO.

[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, Massachusetts Institute of Technology.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., 1989. Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, N. J.; republished by Athena Scientific, Belmont, MA, 1997.

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

[BeT00] Bertsekas, D. P., and Tsitsiklis, J. N., 2000. "Gradient Convergence in Gradient Methods," SIAM J. on Optimization, Vol. 10, pp. 627-642.

[BeY07] Bertsekas, D. P., and Yu, H., 2007. "Solution of Large Systems of Equations Using Approximate Dynamic Programming Methods," LIDS

Report 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.

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

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

[Ber77] Bertsekas, D. P., 1977. "Monotone Mappings with Application in Dynamic Programming," SIAM J. on Control and Optimization, Vol. 15, pp. 438-464.

[Ber82] Bertsekas, D. P., 1982. "Distributed Dynamic Programming," IEEE Trans. Automatic Control, Vol. AC-27, pp. 610-616.

[Ber83] Bertsekas, D. P., 1983. "Asynchronous Distributed Computation of Fixed Points," Math. Programming, Vol. 27, pp. 107-120.

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

[Ber96] Bertsekas, D. P., 1996. Lecture at NSF Workshop on Reinforcement Learning, Hilltop House, Harper's Ferry, N.Y.

[Ber97] Bertsekas, D. P., 1997. "Differential Training of Rollout Policies," Proc. of the 35th Allerton Conference on Communication, Control, and Computing, Allerton Park, Ill.

[Ber99] Bertsekas, D. P., 1999. Nonlinear Programming: 2nd Edition, Athena Scientific, Belmont, MA.

[Ber05a] Bertsekas, D. P., 2005. "Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC," in Fundamental Issues in Control, European J. of Control, Vol. 11.

[Ber05b] Bertsekas, D. P., 2005. "Rollout Algorithms for Constrained Dynamic Programming," Lab. for Information and Decision Systems Report 2646, MIT.

[Ber09a] Bertsekas, D. P., 2009. Convex Optimization Theory, Athena Scientific, Belmont, MA.

[Ber09b] Bertsekas, D. P., 2009. "Projected Equations, Variational Inequalities, and Temporal Difference Methods," Lab. for Information and Decision Systems Report LIDS-P-2808, MIT.

[Ber10a] Bertsekas, D. P., 2010. "Rollout Algorithms for Discrete Opti-

mization: A Survey," Lab. for Information and Decision Systems Report, MIT; to appear in Handbook of Combinatorial Optimization, by D. Zu, and P. Pardalos (eds.), Springer, N. Y.

[Ber10b] Bertsekas, D. P., 2010. "Approximate Policy Iteration: A Survey and Some New Methods," Lab. for Information and Decision Systems Report LIDS-P-2833, MIT; J. of Control Theory and Applications, Vol. 9, pp. 310-335.

[Ber10c] Bertsekas, D. P., 2010. "Pathologies of Temporal Difference Methods in Approximate Dynamic Programming," Proc. 2010 IEEE Conference on Decision and Control.

[Ber10d] Bertsekas, D. P., 2010. "Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey," Lab. for Information and Decision Systems Report LIDS-P-2848, MIT.

[Ber11a] Bertsekas, D. P., 2011. "Temporal Difference Methods for General Projected Equations," IEEE Trans. on Automatic Control, Vol. 56, (to appear).

[Ber11b] Bertsekas, D. P., 2011. "λ-Policy Iteration: A Review and a New Implementation," Lab. for Information and Decision Systems Report LIDS-P-2874, MIT; to appear in *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, by F. Lewis and D. Liu (eds.), IEEE Press Computational Intelligence Series.

[BoM00] Borkar, V. S., and Meyn, S. P., 2000. "The O.D.E. Method for Convergence of Stochastic Approximation and Reinforcement Learning, SIAM J. Control and Optimization, Vol. 38, pp. 447-469.

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

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

[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.

[Bur97] Burgiel, H., 1997. "How to Lose at Tetris," The Mathematical Gazette, Vol. 81, pp. 194-200.

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

[CPS92] Cottle, R. W., Pang, J-S., and Stone, R. E., 1992. The Linear

Complementarity Problem, Academic Press, N. Y.; republished by SIAM in 2009.

[CaC97] Cao, X. R., and Chen, H. F., 1997. "Perturbation Realization Potentials and Sensitivity Analysis of Markov Processes," IEEE Transactions on Automatic Control, Vol. 32, pp. 1382-1393.

[CaW98] Cao, X. R., and Wan, Y. W., 1998. "Algorithms for Sensitivity Analysis of Markov Systems Through Potentials and Perturbation Realization," IEEE Transactions Control Systems Technology, Vol. 6, pp. 482-494.

[Cao99] Cao, X. R., 1999. "Single Sample Path Based Optimization of Markov Chains," J. of Optimization Theory and Applicationa, Vol. 100, pp. 527-548.

[Cao04] Cao, X. R., 2004. "Learning and Optimization from a System Theoretic Perspective," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, (Eds.), IEEE Press, N. Y.

[Cao05] Cao, X. R., 2005. "A Basic Formula for Online Policy Gradient Algorithms," IEEE Transactions on Automatic Control, Vol. 50, pp. 696-699.

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

[ChV06] Choi, D. S., and Van Roy, B., 2006. "A Generalized Kalman Filter for Fixed Point Approximation and Efficient Temporal-Difference Learning," Discrete Event Dynamic Systems, Vol. 16, pp. 207-239.

[DFM09] Desai, V. V., Farias, V. F., and Moallemi, C. C., 2009. "Aproximate Dynamic Programming via a Smoothed Approximate Linear Program, Submitted.

[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.

[DFV03] de Farias, D. P., and Van Roy, B., 2003. "The Linear Programming Approach to Approximate Dynamic Programming," Operations Research, Vol. 51, pp. 850-865.

[DFV04a] de Farias, D. P., and Van Roy, B., 2004. "On Constraint Sampling in the Linear Programming Approach to Approximate Dynamic Programming," Mathematics of Operations Research, Vol. 29, pp. 462-478.

[Day92] Dayan, P., 1992. "The Convergence of TD(λ) for General λ," Machine Learning, Vol. 8, pp. 341-362.

[DeF04] De Farias, D. P., 2004. "The Linear Programming Approach to Approximate Dynamic Programming," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, (Eds.), IEEE Press,

N. Y.

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

[EGW06] Ernst, D., Geurts, P., and Wehenkel, L., 2006. "Tree-Based Batch Mode Reinforcement Learning," Journal of Machine Learning Research, Vol. 6, pp. 503556.

[ELP09] Moazzez-Estanjini, R., Li, K., and Paschalidis, I. C., 2009. "An Actor-Critic Method Using Least Squares Temporal Difference Learning with an Application to Warehouse Management," Proc. of the 48th IEEE Conference on Decision and Control, Shanghai, China, pp. 2564-2569.

[FaV06] Farias, V. F., and Van Roy, B., 2006. "Tetris: A Study of Randomized Constraint Sampling, in Probabilistic and Randomized Methods for Design Under Uncertainty, Springer-Verlag.

[FeS94] Feinberg, E. A., and Shwartz, A., 1994. "Markov Decision Models with Weighted Discounted Criteria," Mathematics of Operations Research, Vol. 19, pp. 1-17.

[FeS04] Ferrari, S., and Stengel, R. F., 2004. "Model-Based Adaptive Critic Designs," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, (Eds.), IEEE Press, N. Y.

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

[FuH94] Fu, M. C., and Hu, 1994. "Smoothed Perturbation Analysis Derivative Estimation for Markov Chains," Oper. Res. Letters, Vol. 41, pp. 241-251.

[GKP03] Guestrin, C. E., Koller, D., Parr, R., and Venkataraman, S., 2003. "Efficient Solution Algorithms for Factored MDPs," J. of Artificial Intelligence Research, Vol. 19, pp. 399-468.

[GLH94] Gurvits, L., Lin, L. J., and Hanson, S. J., 1994. "Incremental Learning of Evaluation Functions for Absorbing Markov Chains: New Methods and Theorems," Preprint.

[GlI89] Glynn, P. W., and Iglehart, D. L., 1989. "Importance Sampling for Stochastic Simulations," Management Science, Vol. 35, pp. 1367-1392.

[Gly87] Glynn, P. W., 1987. "Likelihood Ratio Gradient Estimation: An Overview," Proc. of the 1987 Winter Simulation Conference, pp. 366-375.

[Gor95] Gordon, G. J., 1995. "Stable Function Approximation in Dynamic Programming," in Machine Learning: Proceedings of the Twelfth International Conference, Morgan Kaufmann, San Francisco, CA.

[Gos03] Gosavi, A., 2003. Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning, Springer-Verlag, N. Y.

[Gos04] Gosavi, A., 2004. "Reinforcement Learning for Long-Run Average Cost," European J. of Operational Research, Vol. 155, pp. 654-674.

[GrU04] Grudic, G., and Ungar, L., 2004. "Reinforcement Learning in Large, High-Dimensional State Spaces," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, (Eds.), IEEE Press, N. Y.

[HBK94] Harmon, M. E., Baird, L. C., and Klopf, A. H., 1994. "Advantage Updating Applied to a Differential Game," Presented at NIPS Conf., Denver, Colo.

[HFM05] He, Y., Fu, M. C., and Marcus, S. I., 2005. "A Two-Timescale Simulation-Based Gradient Algorithm for Weighted Cost Markov Decision Processes," Proc. of the 2005 Conf. on Decision and Control, Seville, Spain, pp. 8022-8027.

[Hau00] Hauskrecht, M., 2000. "Value-Function Approximations for Partially Observable Markov Decision Processes, Journal of Artificial Intelligence Research, Vol. 13, pp. 33-95.

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

[He02] He, Y., 2002. Simulation-Based Algorithms for Markov Decision Processes, Ph.D. Thesis, University of Maryland.

[JJS94] Jaakkola, T., Jordan, M. I., and Singh, S. P., 1994. "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms," Neural Computation, Vol. 6, pp. 1185-1201.

[JSJ95] Jaakkola, T., Singh, S. P., and Jordan, M. I., 1995. "Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems," Advances in Neural Information Processing Systems, Vol. 7, pp. 345-352.

[JuP07] Jung, T., and Polani, D., 2007. "Kernelizing LSPE(λ)," in Proc. 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, Honolulu, Hawaii. pp. 338-345.

[KMP06] Keller, P. W., Mannor, S., and Precup, D., 2006. "Automatic Basis Function Construction for Approximate Dynamic Programming and Reinforcement Learning," Proc. of the 23rd ICML, Pittsburgh, Penn.

[Kak02] Kakade, S., 2002. "A Natural Policy Gradient," Proc. Advances in Neural Information Processing Systems, Vancouver, BC, Vol. 14, pp. 1531-1538.

[KoB99] Konda, V. R., and Borkar, V. S., 1999. " Actor-Critic Like Learning Algorithms for Markov Decision Processes," SIAM J. on Control and Optimization, Vol. 38, pp. 94-123.

[KoP00] Koller, K., and Parr, R., 2000. "Policy Iteration for Factored

MDPs," Proc. of the 16th Annual Conference on Uncertainty in AI, pp. 326-334.

[KoT99] Konda, V. R., and Tsitsiklis, J. N., 1999. "Actor-Critic Algorithms," Proc. 1999 Neural Information Processing Systems Conference, Denver, Colorado, pp. 1008-1014.

[KoT03] Konda, V. R., and Tsitsiklis, J. N., 2003. "Actor-Critic Algorithms," SIAM J. on Control and Optimization, Vol. 42, pp. 1143-1166.

[Kon02] Konda, V. R., 2002. Actor-Critic Algorithms, Ph.D. Thesis, Dept. of EECS, M.I.T., Cambridge, MA.

[KuY03] Kushner, H. J., and Yin, G. G., 2003. Stochastic Approximation and Recursive Algorithms and Applications, 2nd Edition, Springer-Verlag, N. Y.

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

[LLL08] Lewis, F. L., Liu, D., and Lendaris, G. G., 2008. Special Issue on Adaptive Dynamic Programming and Reinforcement Learning in Feedback Control, IEEE Transactions on Systems, Man, and Cybernetics, Part B, Vol. 38, No. 4.

[LSS09] Li, Y., Szepesvari, C., and Schuurmans, D., 2009. "Learning Exercise Policies for American Options," Proc. of the Twelfth International Conference on Artificial Intelligence and Statistics, Clearwater Beach, Fla.

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

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

[Liu01] Liu, J. S., 2001. Monte Carlo Strategies in Scientific Computing, Springer, N. Y.

[LoS01] Longstaff, F. A., and Schwartz, E. S., 2001. "Valuing American Options by Simulation: A Simple Least-Squares Approach," Review of Financial Studies, Vol. 14, pp. 113-147.

[MMS06] Menache, I., Mannor, S., and Shimkin, N., 2005. "Basis Function Adaptation in Temporal Difference Reinforcement Learning," Ann. Oper. Res., Vol. 134, pp. 215-238.

[MaT01] Marbach, P., and Tsitsiklis, J. N., 2001. "Simulation-Based Optimization of Markov Reward Processes," IEEE Transactions on Automatic Control, Vol. 46, pp. 191-209.

[MaT03] Marbach, P., and Tsitsiklis, J. N., 2003. "Approximate Gradient

Methods in Policy-Space Optimization of Markov Reward Processes," J. Discrete Event Dynamic Systems, Vol. 13, pp. 111-148.

[Mah96] Mahadevan, S., 1996. "Average Reward Reinforcement Learning: Foundations, Algorithms, and Empirical Results," Machine Learning, Vol. 22, pp. 1-38.

[Mar70] Martinet, B., 1970. "Regularisation d' Inequations Variationnelles par Approximations Successives", Rev. Francaise Inf. Rech. Oper., Vol. 2, pp. 154-159.

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

[MuS08] Munos, R., and Szepesvari, C, 2008. "Finite-Time Bounds for Fitted Value Iteration," Journal of Machine Learning Research, Vol. 1, pp. 815-857.

[Mun03] Munos, R., 2003. "Error Bounds for Approximate Policy Iteration," Proc. 20th International Conference on Machine Learning, pp. 560-567.

[NeB03] Nedić, A., and Bertsekas, D. P., 2003. "Least-Squares Policy Evaluation Algorithms with Linear Function Approximation," J. of Discrete Event Systems, Vol. 13, pp. 79-110.

[OrS02] Ormoneit, D., and Sen, S., 2002. "Kernel-Based Reinforcement Learning," Machine Learning, Vol. 49, pp. 161-178.

[PSD01] Precup, D., Sutton, R. S., and Dasgupta, S., 2001. "Off-Policy Temporal-Difference Learning with Function Approximation," In Proc. 18th Int. Conf. Machine Learning, pp. 417424.

[PWB09] Polydorides, N., Wang, M., and Bertsekas, D. P., 2009. "Approximate Solution of Large-Scale Linear Inverse Problems with Monte Carlo Simulation," Lab. for Information and Decision Systems Report LIDS-P-2822, MIT.

[Pin97] Pineda, F., 1997. "Mean-Field Analysis for Batched TD(λ)," Neural Computation, pp. 1403-1419.

[PoB04] Poupart, P., and Boutilier, C., 2004. "Bounded Finite State Controllers," Advances in Neural Information Processing Systems.

[PoV04] Powell, W. B., and Van Roy, B., 2004. "Approximate Dynamic Programming for High-Dimensional Resource Allocation Problems," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, (Eds.), IEEE Press, N. Y.

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

[Roc76] Rockafellar, R. T.,"Monotone Operators and the Proximal Point

Algorithm," SIAM J. on Control and Optimization, Vol. 14, 1976, pp. 877-898.

[RuK04] Rubinstein, R. Y., and Kroese, D. P., 2004. The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Springer, N. Y.

[RuK08] Rubinstein, R. Y., and Kroese, D. P., 2008. Simulation and the Monte Carlo Method, 2nd Edition, J. Wiley, N. Y.

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

[SDG09] Simao, D. G., Day, S., George, A. P., Gifford, T., Nienow, J., and Powell, W. B., 2009. "An Approximate Dynamic Programming Algorithm for Large-Scale Fleet Management: A Case Application," Transportation Science, Vol. 43, pp. 178197.

[SJJ94] Singh, S. P., Jaakkola, T., and Jordan, M. I., 1994. "Learning without State-Estimation in Partially Observable Markovian Decision Processes," Proceedings of the Eleventh Machine Learning Conference, pp. 284-292.

[SJJ95] Singh, S. P., Jaakkola, T., and Jordan, M. I., 1995. "Reinforcement Learning with Soft State Aggregation," in Advances in Neural Information Processing Systems 7, MIT Press, Cambridge, MA.

[SMS99] Sutton, R. S., McAllester, D., Singh, S. P., and Mansour, Y., 1999. "Policy Gradient Methods for Reinforcement Learning with Function Approximation," Proc. 1999 Neural Information Processing Systems Conference, Denver, Colorado.

[SYL04] Si, J., Yang, L., and Liu, D., 2004. "Direct Neural Dynamic Programming," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, (Eds.), IEEE Press, N. Y.

[Saa03] Saad, Y., 2003. Iterative Methods for Sparse Linear Systems, SIAM, Phila., Pa.

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

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

[ScS85] Schweitzer, P. J., and Seidman, A., 1985. "Generalized Polynomial Approximations in Markovian Decision Problems," J. Math. Anal. and Appl., Vol. 110, pp. 568-582.

[Sch93] Schwartz, A., 1993. "A Reinforcement Learning Method for Maxi-

mizing Undiscounted Rewards," Proceedings of the Tenth Machine Learning Conference, pp. 298-305.

[Sch07] Scherrer, B., 2007. "Performance Bounds for Lambda Policy Iteration," Technical Report 6348, INRIA.

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

[Sha53] Shapley, L. S., 1953. "Stochastic Games," Proc. Nat. Acad. Sci. U.S.A., Vol. 39.

[Sin94] Singh, S. P., 1994. "Reinforcement Learning Algorithms for Average-Payoff Markovian Decision Processes," Proc. of 12th National Conference on Artificial Intelligence, pp. 202-207.

[Str09] Strang, G., 2009. Linear Algebra and its Applications, Wellesley Cambridge Press, Welleslay, MA.

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

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

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

[SzS04] Szepesvari, C., and Smart, W. D., 2004. "Interpolation-Based Q-Learning," Proc. of 21st International Conf. on Machine Learning, Banff, Ca.

[Sze09] Szepesvari, C., 2009. "Reinforcement Learning Algorithms for MDPs," Dept. of Computing Science Report TR09-13, University of Alberta, Ca.

[Tes92] Tesauro, G., 1992. "Practical Issues in Temporal Difference Learning," Machine Learning, Vol. 8, pp. 257-277.

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

[ThS10a] Thiery, C., and Scherrer, B., 2010. "Least-Squares λ-Policy Iteration: Bias-Variance Trade-off in Control Problems," in ICML'10: Proc. of the 27th Annual International Conf. on Machine Learning.

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

[ToS06] Toussaint, M., and Storkey, A. 2006. "Probabilistic Inference for Solving Discrete and Continuous State Markov Decision Processes," in

Proc. of the 23nd ICML, pp. 945-952.

[TrB97] Trefethen, L. N., and Bau, D., 1997. Numerical Linear Algebra, SIAM, Phila., PA.

[TsV96] Tsitsiklis, J. N., and Van Roy, B., 1996. "Feature-Based Methods for Large-Scale Dynamic Programming," Machine Learning, Vol. 22, pp. 59-94.

[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.

[TsV99a] Tsitsiklis, J. N., and Van Roy, B., 1999. "Average Cost Temporal-Difference Learning," Automatica, Vol. 35, pp. 1799-1808.

[TsV99b] Tsitsiklis, J. N., and Van Roy, B., 1999. "Optimal Stopping of Markov Processes: Hilbert Space Theory, Approximation Algorithms, and an Application to Pricing Financial Derivatives," IEEE Transactions on Automatic Control, Vol. 44, pp. 1840-1851.

[TsV01] Tsitsiklis, J. N., and Van Roy, B., 2001. "Regression Methods for Pricing Complex American-Style Options," IEEE Trans. on Neural Networks, Vol. 12, pp. 694-703.

[TsV02] Tsitsiklis, J. N., and Van Roy, B., 2002. "On Average Versus Discounted Reward Temporal–Difference Learning," Machine Learning, Vol. 49, pp. 179-191.

[Tsi94] Tsitsiklis, J. N., 1994. "Asynchronous Stochastic Approximation and Q-Learning," Machine Learning, Vol. 16, pp. 185-202.

[VBL07] Van Roy, B., Bertsekas, D. P., Lee, Y., and Tsitsiklis, J. N., 1997. "A Neuro-Dynamic Programming Approach to Retailer Inventory Management," Proc. of the IEEE Conference on Decision and Control; based on a more extended Lab. for Information and Decision Systems Report, MIT, Nov. 1996.

[Van95] Van Roy, B., 1995. "Feature-Based Methods for Large Scale Dynamic Programming," Lab. for Info. and Decision Systems Report LIDS-TH-2289, Massachusetts Institute of Technology, Cambridge, MA.

[Van98] Van Roy, B., 1998. Learning and Value Function Approximation in Complex Decision Processes, Ph.D. Thesis, Dept. of EECS, MIT, Cambridge, MA.

[Van06] Van Roy, B., 2006. "Performance Loss Bounds for Approximate Value Iteration with State Aggregation," Mathematics of Operations Research, Vol. 31, pp. 234-244.

[VeR06] Verma, R., and Rao, R. P. N., 2006. "Planning and Acting in Uncertain Environments Using Probabilistic Inference," in Proc. of IEEE/RSJ

Intern. Conf. on Intelligent Robots and Systems.

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

[WaB92] Watkins, C. J. C. H., and Dayan, P., 1992. "Q-Learning," Machine Learning, Vol. 8, pp. 279-292.

[Wat89] Watkins, C. J. C. H., Learning from Delayed Rewards, Ph.D. Thesis, Cambridge Univ., England.

[WeB99] Weaver, L., and Baxter, J., 1999. "Reinforcement Learning From State and Temporal Differences," Tech. Report, Department of Computer Science, Australian National University.

[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.
(See http://web.mit.edu/dimitrib/www/Williams-Baird-Counterexample.pdf for a description of this example in a format that is adapted to our context in this chapter.)

[Wil92] Williams, R. J., 1992. "Simple Statistical Gradient Following Algorithms for Connectionist Reinforcement Learning," Machine Learning, Vol. 8, pp. 229-256.

[YaL08] Yao, H., and Liu, Z.-Q., 2008. "Preconditioned Temporal Difference Learning," Proc. of the 25th ICML, Helsinki, Finland.

[YuB04] Yu, H., and Bertsekas, D. P., 2004. "Discretized Approximations for POMDP with Average Cost," Proc. of the 20th Conference on Uncertainty in Artificial Intelligence, Banff, Canada.

[YuB06a] Yu, H., and Bertsekas, D. P., 2006. "On Near-Optimality of the Set of Finite-State Controllers for Average Cost POMDP," Lab. for Information and Decision Systems Report 2689, MIT; Mathematics of Operations Research, Vol. 33, pp. 1-11, 2008.

[YuB06b] Yu, H., and Bertsekas, D. P., 2006. "Convergence Results for Some Temporal Difference Methods Based on Least Squares," Lab. for Information and Decision Systems Report 2697, MIT; also in IEEE Transactions on Aut. Control, Vol. 54, 2009, pp. 1515-1531.

[YuB07] Yu, H., and Bertsekas, D. P., 2007. "A Least Squares Q-Learning Algorithm for Optimal Stopping Problems," Lab. for Information and Decision Systems Report 2731, MIT; also in Proc. European Control Conference 2007, Kos, Greece.

[YuB08] Yu, H., and Bertsekas, D. P., 2008. "Error Bounds for Approximations from Projected Linear Equations," Lab. for Information and Decision

Systems Report LIDS-P-2797, MIT, July 2008; Mathematics of Operations Research, Vol. 35, 2010, pp. 306-329.

[YuB09] Yu, H., and Bertsekas, D. P., 2009. "Basis Function Adaptation Methods for Cost Approximation in MDP," Proceedings of 2009 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2009), Nashville, Tenn.

[YuB11] Yu, H., and Bertsekas, D. P., 2011. "On Boundedness of Q-Learning Iterates for Stochastic Shortest Path Problems," Lab. for Information and Decision Systems Report LIDS-P-2859, MIT, March 2011.

[Yu05] Yu, H., 2005. "A Function Approximation Approach to Estimation of Policy Gradient for POMDP with Structured Policies," Proc. of the 21st Conference on Uncertainty in Artificial Intelligence, Edinburgh, Scotland.

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

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

[ZFM10] Zhou, E., Fu, M. C., and Marcus, S. I., 2010. "Solving Continuous-state POMDPs via Density Projection," IEEE Trans. Automatic Control, Vol. AC-55, pp. 11011116.

[ZhH01] Zhou, R., and Hansen, E. A., 2001. "An Improved Grid-Based Approximation Algorithm for POMDPs," In Int. J. Conf. Artificial Intelligence.

[ZhL97] Zhang, N. L., and Liu, W., 1997. "A Model Approximation Scheme for Planning in Partially Observable Stochastic Domains," J. Artificial Intelligence Research, Vol. 7, pp. 199-230.