3

An Abstract View of Reinforcement Learning

Contents

In this chapter we will use geometric constructions to obtain insight into Bellman's equation, the value and policy iteration algorithms, approximation in value space, and some of the properties of the corresponding one-step or multistep lookahead policy $\tilde{\mu}$. To understand these constructions, we need an abstract notational framework that is based on the operators that are involved in the Bellman equations.

3.1 BELLMAN OPERATORS

We denote by TJ the function of x that appears in the right-hand side of Bellman's equation. Its value at state x is given by

$$(TJ)(x) = \min_{u \in U(x)} E\Big\{ g(x,u,w) + \alpha J\big(f(x,u,w)\big) \Big\}, \qquad \text{for all } x. \tag{3.1}$$

Also for each policy μ, we introduce the corresponding function $T_\mu J$, which has value at x given by

$$(T_\mu J)(x) = E\Big\{ g\big(x,\mu(x),w\big) + \alpha J\big(f(x,\mu(x),w)\big) \Big\}, \qquad \text{for all } x. \tag{3.2}$$

Thus T and T_μ can be viewed as operators (broadly referred to as the *Bellman operators*), which map functions J to other functions (TJ or $T_\mu J$, respectively).†

An important property of the operators T and T_μ is that they are *monotone*, in the sense that if J and J' are two functions of x such that

$$J(x) \geq J'(x), \qquad \text{for all } x,$$

then we have

$$(TJ)(x) \geq (TJ')(x), \qquad (T_\mu J)(x) \geq (T_\mu J')(x), \qquad \text{for all } x \text{ and } \mu. \tag{3.3}$$

This monotonicity property is evident from Eqs. (3.1) and (3.2), where the values of J are multiplied by nonnegative numbers.

† Within the context of this work, the functions J on which T and T_μ operate will be real-valued functions of x, which we denote by $J \in R(X)$. We will assume throughout that the expected values in Eqs. (3.1) and (3.2) are well-defined and finite when J is real-valued. This implies that $T_\mu J$ will also be real-valued functions of x. On the other hand $(TJ)(x)$ may take the value $-\infty$ because of the minimization in Eq. (3.1). We allow this possibility, although our illustrations will primarily depict the case where TJ is real-valued. Note that the general theory of abstract DP is developed with the use of extended real-valued functions; see [Ber22a].

Another important property is that the Bellman operator T_μ is *linear*, in the sense that it has the form $T_\mu J = G + A_\mu J$, where $G \in R(X)$ is some function and $A_\mu : R(X) \mapsto R(X)$ is an operator such that for any functions J_1, J_2, and scalars γ_1, γ_2, we have†

$$A_\mu(\gamma_1 J_1 + \gamma_2 J_2) = \gamma_1 A_\mu J_1 + \gamma_2 A_\mu J_2.$$

Moreover, from the definitions (3.1) and (3.2), we have

$$(TJ)(x) = \min_{\mu \in \mathcal{M}} (T_\mu J)(x), \qquad \text{for all } x,$$

where $\mathcal{M}$ is the set of stationary policies. This is true because for any policy μ, there is no coupling constraint between the controls $\mu(x)$ and $\mu(x')$ that correspond to two different states x and x'. It follows that $(TJ)(x)$ *is a concave function of* J *for every* x (the pointwise minimum of linear functions is a concave function). This will be important for our interpretation of one-step and multistep lookahead minimization as a Newton iteration for solving the Bellman equation $J = TJ$.

Example 3.1.1 (A Two-State and Two-Control Example)

Assume that there are two states 1 and 2, and two controls u and v. Consider the policy μ that applies control u at state 1 and control v at state 2. Then the operator T_μ takes the form

$$(T_\mu J)(1) = \sum_{y=1}^{2} p_{1y}(u)\big(g(1,u,y) + \alpha J(y)\big), \tag{3.4}$$

$$(T_\mu J)(2) = \sum_{y=1}^{2} p_{2y}(v)\big(g(2,v,y) + \alpha J(y)\big), \tag{3.5}$$

where $p_{xy}(u)$ and $p_{xy}(v)$ are the probabilities that the next state will be y, when the current state is x, and the control is u or v, respectively. Clearly, $(T_\mu J)(1)$ and $(T_\mu J)(2)$ are linear functions of J. Also the operator T of the Bellman equation $J = TJ$ takes the form

$$\begin{aligned}(TJ)(1) = \min\Bigg[&\sum_{y=1}^{2} p_{1y}(u)\big(g(1,u,y) + \alpha J(y)\big), \\ &\sum_{y=1}^{2} p_{1y}(v)\big(g(1,v,y) + \alpha J(y)\big)\Bigg],\end{aligned} \tag{3.6}$$

† An operator T_μ with this property is often called "affine," but in this work we just call it "linear." Also we use abbreviated notation to express pointwise equalities and inequalities, so that we write $J = J'$ or $J \geq J'$ to express the fact that $J(x) = J'(x)$ or $J(x) \geq J'(x)$, for all x, respectively.

$$(TJ)(2) = \min\left[\sum_{y=1}^{2} p_{2y}(u)\big(g(2,u,y) + \alpha J(y)\big), \sum_{y=1}^{2} p_{2y}(v)\big(g(2,v,y) + \alpha J(y)\big)\right]. \tag{3.7}$$

Thus, $(TJ)(1)$ and $(TJ)(2)$ are concave and piecewise linear as functions of the two-dimensional vector J (with two pieces; more generally, as many linear pieces as the number of controls). This concavity property holds in general since $(TJ)(x)$ is the minimum of a collection of linear functions of J, one for each $u \in U(x)$. Figure 3.1.1 illustrates $(T_\mu J)(1)$ for the cases where $\mu(1) = u$ and $\mu(1) = v$, $(T_\mu J)(2)$ for the cases where $\mu(2) = u$ and $\mu(2) = v$, $(TJ)(1)$, and $(TJ)(2)$, as functions of $J = \big(J(1), J(2)\big)$.

Critical properties from the DP point of view are whether T and T_μ have fixed points; equivalently, whether the Bellman equations $J = TJ$ and $J = T_\mu J$ have solutions within the class of real-valued functions, and whether the set of solutions includes J^* and J_μ, respectively. It may thus be important to verify that T or T_μ are contraction mappings. This is true for example in the benign case of discounted problems with bounded cost per stage. However, for undiscounted problems, asserting the contraction property of T or T_μ may be more complicated, and even impossible; the abstract DP book [Ber22a] deals extensively with such questions, and related issues regarding the solution sets of the Bellman equations.

Geometrical Interpretations

We will now interpret the Bellman operators geometrically, starting with T_μ. Figure 3.1.2 illustrates its form. Note here that the functions J and $T_\mu J$ are multidimensional. They have as many scalar components $J(x)$ and $(T_\mu J)(x)$, respectively, as there are states x, but they can only be shown projected onto one dimension. The function $T_\mu J$ for each policy μ is linear. The cost function J_μ satisfies $J_\mu = T_\mu J_\mu$, so it is obtained from the intersection of the graph of $T_\mu J$ and the 45 degree line, when J_μ is real-valued. Later we will interpret the case where J_μ is not real-valued as the system being unstable under μ [we have $J_\mu(x) = \infty$ for some initial states x].

The form of the Bellman operator T is illustrated in Fig. 3.1.3. Again the functions J, J^*, TJ, $T_\mu J$, etc, are multidimensional, but they are shown projected onto one dimension (alternatively they are illustrated for a system with a single state, plus possibly a termination state). The Bellman equation $J = TJ$ may have one or many real-valued solutions. It may also have no real-valued solution in exceptional situations, as we will discuss later (see Section 3.8). The figure assumes a unique real-valued solution of the Bellman equations $J = TJ$ and $J = T_\mu J$, which is true if T and T_μ are contraction mappings, as is the case for discounted problems with

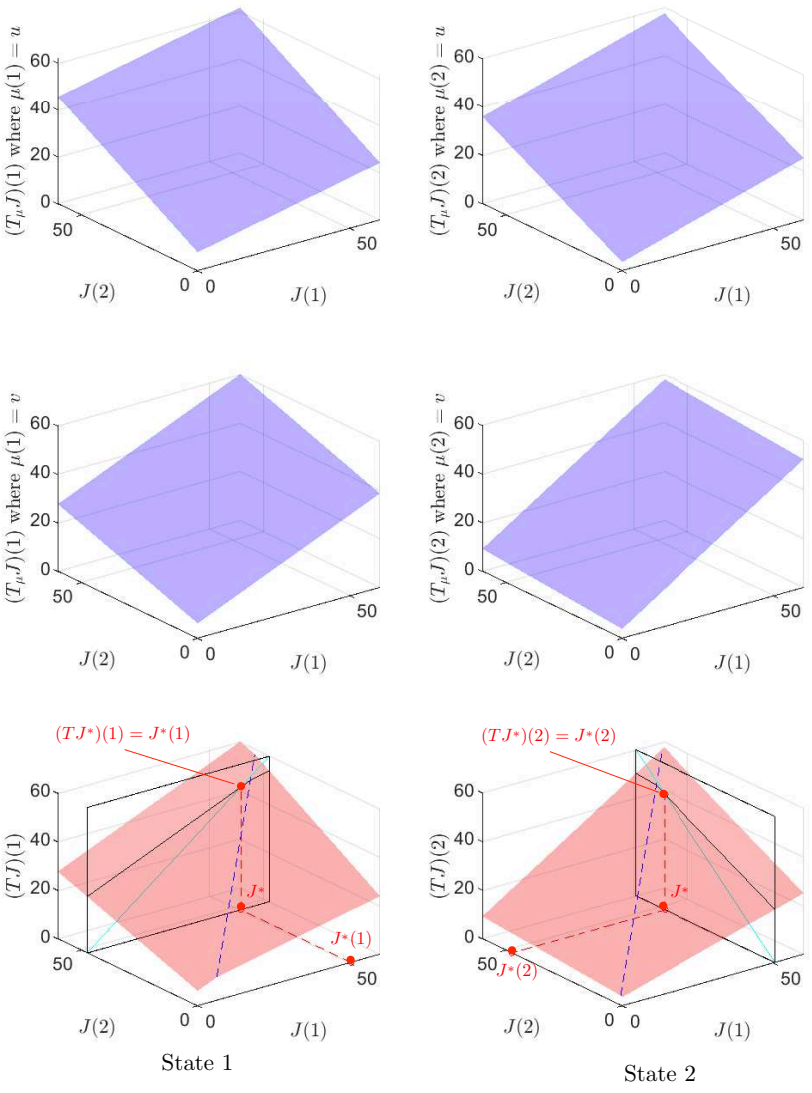

Figure 3.1.1 Geometric illustrations of the Bellman operators T_μ and T for states 1 and 2 in Example 3.1.1; cf. Eqs. (3.4)-(3.7). The problem's transition probabilities are: $p_{11}(u) = 0.3$, $p_{12}(u) = 0.7$, $p_{21}(u) = 0.4$, $p_{22}(u) = 0.6$, $p_{11}(v) = 0.6$, $p_{12}(v) = 0.4$, $p_{21}(v) = 0.9$, $p_{22}(v) = 0.1$. The stage costs are $g(1, u, 1) = 3$, $g(1, u, 2) = 10$, $g(2, u, 1) = 0$, $g(2, u, 2) = 6$, $g(1, v, 1) = 7$, $g(1, v, 2) = 5$, $g(2, v, 1) = 3$, $g(2, v, 2) = 12$. The discount factor is $\alpha = 0.9$, and the optimal costs are $J^*(1) = 50.59$ and $J^*(2) = 47.41$. The optimal policy is $\mu^*(1) = v$ and $\mu^*(2) = u$. The figure also shows two one-dimensional slices of T that are parallel to the $J(1)$ and $J(2)$ axes and pass through J^*.

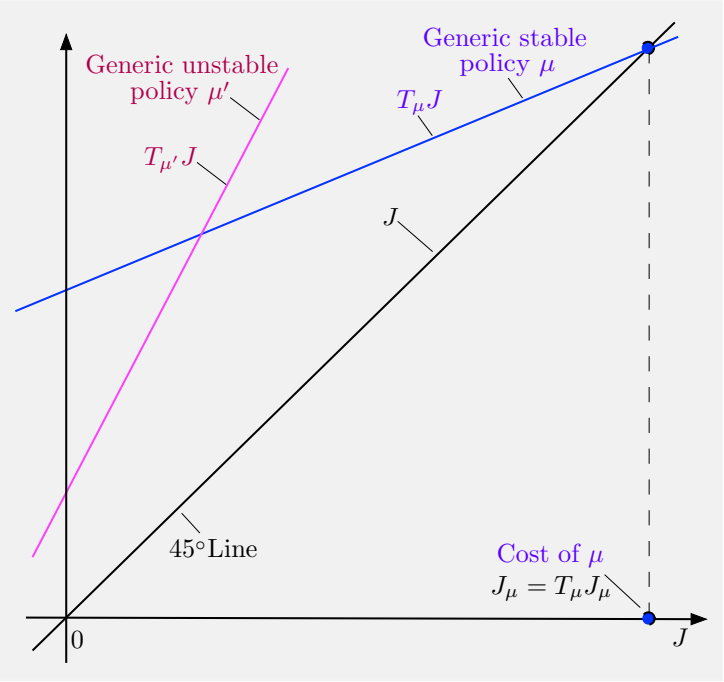

Figure 3.1.2 Geometric interpretation of the linear Bellman operator T_μ and the corresponding Bellman equation. The graph of T_μ is a plane in the space $R(X) \times R(X)$, and when projected on a one-dimensional plane that corresponds to a single state and passes through J_μ, it becomes a line. Then there are three cases:

(a) The line has slope less than 45 degrees, so it intersects the 45-degree line at a unique point, which is equal to J_μ, the solution of the Bellman equation $J = T_\mu J$. This is true if T_μ is a contraction mapping, as is the case for discounted problems with bounded cost per stage.

(b) The line has slope greater than 45 degrees. Then it intersects the 45-degree line at a unique point, which is a solution of the Bellman equation $J = T_\mu J$, but is not equal to J_μ. Then J_μ is not real-valued; we will call such μ *unstable* in Section 3.2.

(c) The line has slope exactly equal to 45 degrees. This is an exceptional case where the Bellman equation $J = T_\mu J$ has an infinite number of real-valued solutions or no real-valued solution at all; we will provide examples where this occurs in Section 3.8.

bounded cost per stage. Otherwise, these equations may have no solution or multiple solutions within the class of real-valued functions (see Section 3.8). The equation $J = TJ$ typically has J^* as a solution, but may have more than one solution in cases where either $\alpha = 1$, or $\alpha < 1$ and the cost per stage is unbounded.

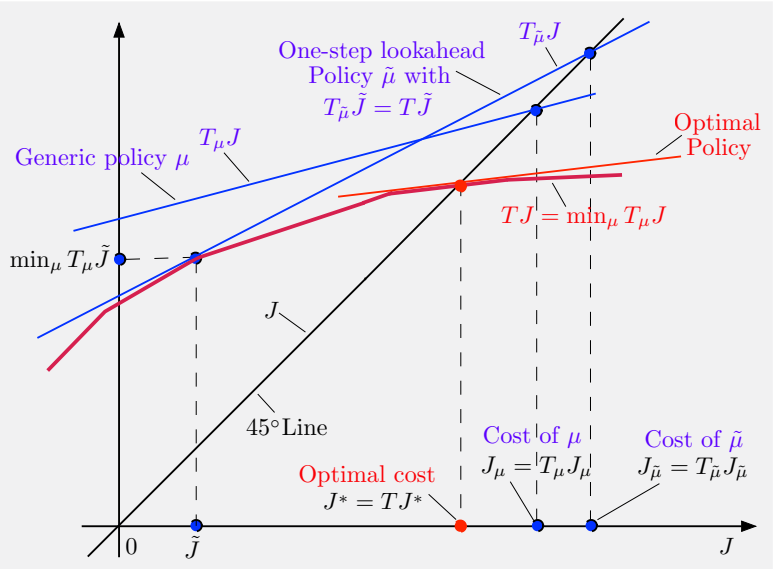

Figure 3.1.3 Geometric interpretation of the Bellman operator T, and the corresponding Bellman equation. For a fixed x, the function $(TJ)(x)$ can be written as $\min_\mu (T_\mu J)(x)$, so it is concave as a function of J. The optimal cost function J^* satisfies $J^* = TJ^*$, so it is obtained from the intersection of the graph of TJ and the 45 degree line shown, assuming J^* is real-valued.

Note that the graph of T lies below the graph of every operator T_μ, and is in fact obtained as the lower envelope of the graphs of T_μ as μ ranges over the set of policies $\mathcal{M}$. In particular, for any given function $\tilde{J}$, for every x, the value $(T\tilde{J})(x)$ is obtained by finding a support hyperplane/subgradient of the graph of the concave function $(TJ)(x)$ at $J = \tilde{J}$, as shown in the figure. This support hyperplane is defined by the control $\mu(x)$ of a policy $\tilde{\mu}$ that attains the minimum of $(T_\mu \tilde{J})(x)$ over μ:

$$\tilde{\mu}(x) \in \arg\min_{\mu \in \mathcal{M}} (T_\mu \tilde{J})(x)$$

(there may be multiple policies attaining this minimum, defining multiple support hyperplanes).

Example 3.1.2 (A Two-State and Infinite Controls Problem)

Let us consider the mapping T for a problem that involves two states, 1 and 2, but an infinite number of controls. In particular, the control space at both states is the unit interval, $U(1) = U(2) = [0, 1]$. Here $(TJ)(1)$ and $(TJ)(2)$ are given by

$$(TJ)(1) = \min_{u \in [0,1]} \Big\{ g_1 + r_{11}u^2 + r_{12}(1-u)^2 + \alpha u J(1) + \alpha(1-u)J(2) \Big\},$$

$$(TJ)(2) = \min_{u \in [0,1]} \Big\{ g_2 + r_{21}u^2 + r_{22}(1-u)^2 + \alpha u J(1) + \alpha(1-u)J(2) \Big\}.$$

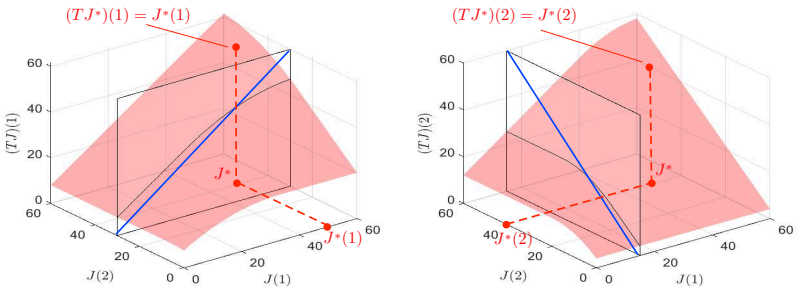

Figure 3.1.4 Illustration of the Bellman operator T for states 1 and 2 in Example 3.1.2. The parameter values are $g_1 = 5$, $g_2 = 3$, $r_{11} = 3$, $r_{12} = 15$, $r_{21} = 9$, $r_{22} = 1$, and the discount factor is $\alpha = 0.9$. The optimal costs are $J^*(1) = 49.7$ and $J^*(2) = 40.0$, and the optimal policy is $\mu^*(1) = 0.59$ and $\mu^*(2) = 0$. The figure also shows the two one-dimensional slices of the operators at $J(1) = 15$ and $J(2) = 30$ that are parallel to the $J(1)$ and $J(2)$ axes.

The control u at each state $x = 1, 2$ has the meaning of a probability that we must select at that state. In particular, we control the probabilities u and $(1-u)$ of moving to states $y = 1$ and $y = 2$, at a control cost that is quadratic in u and $(1-u)$, respectively. For this problem $(TJ)(1)$ and $(TJ)(2)$ can be calculated in closed form, so they are easy to plot and understand. They are piecewise quadratic, unlike the corresponding plots of Fig. 3.1.1, which are piecewise linear; see Fig. 3.1.4.

Visualization of Value Iteration

The operator notation simplifies algorithmic descriptions, derivations, and proofs related to DP. For example, we can write the VI algorithm in the compact form

$$J_{k+1} = TJ_k, \qquad k = 0, 1, \ldots,$$

as illustrated in Fig. 3.1.5. Moreover, the VI algorithm for a given policy μ can be written as

$$J_{k+1} = T_\mu J_k, \qquad k = 0, 1, \ldots,$$

and it can be similarly interpreted, except that the graph of the function $T_\mu J$ is linear. Also we will see shortly that there is a similarly compact description for the policy iteration algorithm.

To keep the presentation simple, we will focus our attention on the abstract DP framework as it applies to the optimal control problems of Section 2.1. In particular, we will assume without further mention that T *and* T_μ *have the monotonicity property (3.3), that* $T_\mu J$ *is linear for all* μ*, and*

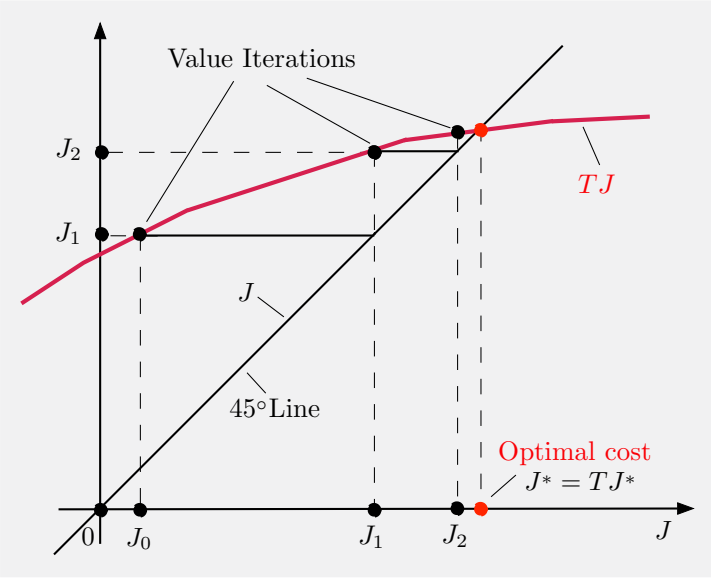

Figure 3.1.5 Geometric interpretation of the VI algorithm $J_{k+1} = TJ_k$, starting from some initial function J_0. Successive iterates are obtained through the staircase construction shown in the figure. The VI algorithm $J_{k+1} = T_\mu J_k$ for a given policy μ can be similarly interpreted, except that the graph of the function $T_\mu J$ is linear.

that (as a consequence) the component $(TJ)(x)$ *is concave as a function of* J *for every state* x. We note, however, that the abstract notation facilitates the extension of the infinite horizon DP theory to models beyond the ones that we discuss in this work. Such models include semi-Markov problems, minimax control problems, risk sensitive problems, Markov games, and others (see the DP textbook [Ber12], and the abstract DP monograph [Ber22a]).

3.2 APPROXIMATION IN VALUE SPACE AND NEWTON'S METHOD

Let us now consider approximation in value space and an abstract geometric interpretation, first provided in the author's book [Ber20a]. By using the operators T and T_μ, for a given $\tilde{J}$, a one-step lookahead policy $\tilde{\mu}$ is characterized by the equation $T_{\tilde{\mu}}\tilde{J} = T\tilde{J}$, or equivalently

$$\tilde{\mu}(x) \in \arg\min_{u \in U(x)} E\Big\{g(x,u,w) + \alpha \tilde{J}\big(f(x,u,w)\big)\Big\}, \tag{3.8}$$

as in Fig. 3.2.1. Furthermore, this equation implies that the graph of $T_{\tilde{\mu}}J$ just touches the graph of TJ at $\tilde{J}$, as shown in the figure.

In mathematical terms, for each state $x \in X$, the hyperplane $H_{\tilde{\mu}}(x) \in R(X) \times \Re$

$$H_{\tilde{\mu}}(x) = \big\{(J, \xi) \mid (T_{\tilde{\mu}}J)(x) = \xi\big\}, \tag{3.9}$$

supports from above the hypograph of the concave function $(TJ)(x)$, i.e., the convex set

$$\big\{(J, \xi) \mid (TJ)(x) \geq \xi\big\}.$$

The point of support is $\big(\tilde{J}, (T_{\tilde{\mu}}\tilde{J})(x)\big)$, and relates the function $\tilde{J}$ with the corresponding one-step lookahead minimization policy $\tilde{\mu}$, the one that satisfies $T_{\tilde{\mu}}\tilde{J} = T\tilde{J}$. The hyperplane $H_{\tilde{\mu}}(x)$ of Eq. (3.9) defines a subgradient of $(TJ)(x)$ at $\tilde{J}$. Note that the one-step lookahead policy $\tilde{\mu}$ need not be unique, since T need not be differentiable, so there may be multiple hyperplanes of support at $\tilde{J}$. Still this construction shows that the linear operator $T_{\tilde{\mu}}$ is a linearization of the operator T at the point $\tilde{J}$ (pointwise for each x).

Equivalently, for every $x \in X$, the linear scalar equation $J(x) = (T_{\tilde{\mu}}J)(x)$ is a linearization of the nonlinear equation $J(x) = (TJ)(x)$ at the point $\tilde{J}$. Consequently, the linear operator equation $J = T_{\tilde{\mu}}J$ is a linearization of the equation $J = TJ$ at $\tilde{J}$, and its solution, $J_{\tilde{\mu}}$, can be viewed as the result of a Newton iteration at the point $\tilde{J}$ (here we adopt an expanded view of the Newton iteration that applies to possibly nondifferentiable fixed point equations; see the Appendix). In summary, *the Newton iterate at $\tilde{J}$ is $J_{\tilde{\mu}}$*, the solution of the linearized equation $J = T_{\tilde{\mu}}J$.†

† The classical Newton's method for solving a fixed point problem of the form $y = G(y)$, where y is an n-dimensional vector, operates as follows: At the current iterate y_k, we linearize G and find the solution y_{k+1} of the corresponding linear fixed point problem. Assuming G is differentiable, the linearization is obtained by using a first order Taylor expansion:

$$y_{k+1} = G(y_k) + \frac{\partial G(y_k)}{\partial y}(y_{k+1} - y_k),$$

where $\partial G(y_k)/\partial y$ is the $n \times n$ Jacobian matrix of G evaluated at the vector y_k. The most commonly given convergence rate property of Newton's method is *quadratic convergence*. It states that near the solution y^*, we have

$$\|y_{k+1} - y^*\| = O\big(\|y_k - y^*\|^2\big),$$

where $\|\cdot\|$ is the Euclidean norm, and holds assuming the Jacobian matrix exists, is invertible, and is Lipschitz continuous (see the books by Ortega and Rheinboldt [OrR70], and by the author [Ber16], Section 1.4).

There are well-studied extensions of Newton's method that are based on solving a linearized system at the current iterate, but relax the differentiability requirement through alternative requirements of piecewise differentiability, B-differentiability, and semi-smoothness, while maintaining the superlinear convergence property of the method. In particular, the quadratic rate of convergence

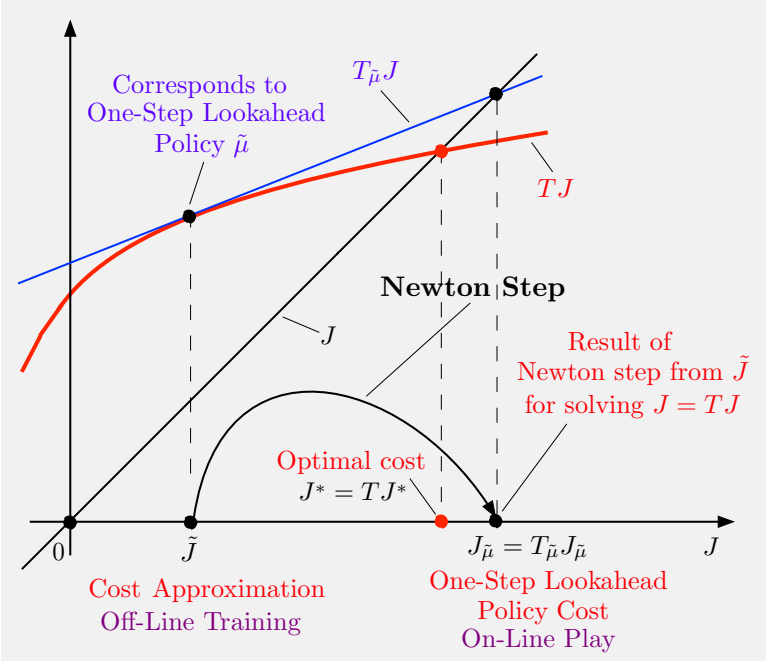

Figure 3.2.1 Geometric interpretation of approximation in value space and the one-step lookahead policy $\tilde{\mu}$ as a step of Newton's method [cf. Eq. (3.8)]. Given $\tilde{J}$, we find a policy $\tilde{\mu}$ that attains the minimum in the relation

$$T\tilde{J} = \min_{\mu} T_{\mu}\tilde{J}.$$

This policy satisfies $T\tilde{J} = T_{\tilde{\mu}}\tilde{J}$, so the graph of TJ and $T_{\tilde{\mu}}J$ touch at $\tilde{J}$, as shown. It may not be unique. Because TJ has concave components, the equation $J = T_{\tilde{\mu}}J$ is the linearization of the equation $J = TJ$ at $\tilde{J}$ [for each x, the hyperplane $H_{\tilde{\mu}}(x)$ of Eq. (3.9) defines a subgradient of $(TJ)(x)$ at $\tilde{J}$]. The linearized equation is solved at the typical step of Newton's method to provide the next iterate, which is just $J_{\tilde{\mu}}$.

The structure of the Bellman operators (3.1) and (3.2), with their monotonicity and concavity properties, tends to enhance the convergence and the rate of convergence properties of Newton's method, even in the absence of differentiability, as evidenced by the favorable Newton-related convergence analysis of PI, and the extensive favorable experience with rollout, PI, and MPC. In fact, the role of monotonicity and concavity in affecting the convergence properties of Newton's method has been addressed

result for differentiable G of Prop. 1.4.1 of the book [Ber16] admits a straightforward and intuitive extension to piecewise differentiable G, given in the paper [KoS86]; see the Appendix, which contains references to the literature.

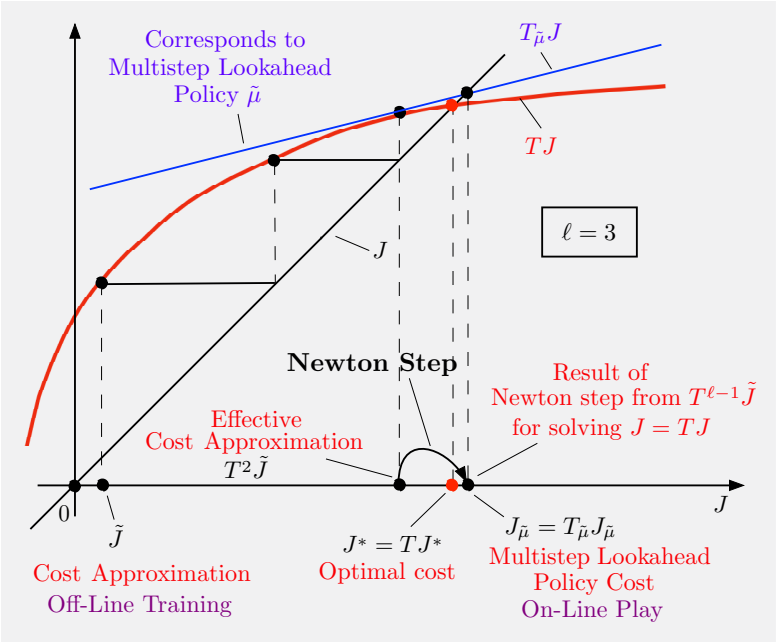

Figure 3.2.2 Geometric interpretation of approximation in value space with ℓ-step lookahead (in this figure $\ell = 3$). It is the same as approximation in value space with one-step lookahead using $T^{\ell-1}\tilde{J}$ as cost approximation. It can be viewed as a Newton step at the point $T^{\ell-1}\tilde{J}$, the result of $\ell - 1$ value iterations applied to $\tilde{J}$. Note that as ℓ increases the cost function $J_{\tilde{\mu}}$ of the ℓ-step lookahead policy $\tilde{\mu}$ approaches more closely the optimal J^*, and that $\lim_{\ell\to\infty} J_{\tilde{\mu}} = J^*$.

in the mathematical literature.†

As noted earlier, approximation in value space with ℓ-step lookahead using $\tilde{J}$ is the same as approximation in value space with one-step lookahead using the $(\ell-1)$-fold operation of T on $\tilde{J}$, $T^{\ell-1}\tilde{J}$. Thus it can be interpreted as a Newton step starting from $T^{\ell-1}\tilde{J}$, the result of $\ell - 1$ value iterations applied to $\tilde{J}$. This is illustrated in Fig. 3.2.2.‡

† See the papers by Ortega and Rheinboldt [OrR67], and Vandergraft [Van67], the books by Ortega and Rheinboldt [OrR70], and Argyros [Arg08], and the references cited there. In this connection, it is worth noting that in the case of Markov games, where the concavity property does not hold, the PI method may oscillate, as shown by Pollatschek and Avi-Itzhak [PoA69], and needs to be modified to restore its global convergence; see the author's paper [Ber21c], and the references cited there.

‡ We note that several variants of Newton's method that involve combinations of first-order iterative methods, such as the Gauss-Seidel and Jacobi algorithms, and Newton's method, are well-known in numerical analysis. They belong to the general family of *Newton-SOR methods* (SOR stands for "succes-

Let us also note that ℓ-step lookahead minimization involves ℓ successive VI iterations, but *only the first of these iterations has a Newton step interpretation*. As an example, consider two-step lookahead minimization with a terminal cost approximation $\tilde{J}$. The second step minimization is a VI that starts from $\tilde{J}$ to produce $T\tilde{J}$. The first step minimization is a VI that starts from $T\tilde{J}$ to produce $T^2\tilde{J}$, but it also does something else that is more significant: *It produces a two-step lookahead minimization policy $\tilde{\mu}$ through $T_{\tilde{\mu}}(T\tilde{J}) = T(T\tilde{J})$, and the step from $T\tilde{J}$ to $J_{\tilde{\mu}}$ (the cost function of $\tilde{\mu}$) is the Newton step*. Thus, there is only one policy produced (i.e., $\tilde{\mu}$) and only one Newton step (from $T\tilde{J}$ to $J_{\tilde{\mu}}$). In the case of one-step lookahead minimization, the Newton step starts from $\tilde{J}$ and ends at $J_{\tilde{\mu}}$. Similarly, in the case of ℓ-step lookahead minimization, the first step of the lookahead is the Newton step (from $T^{\ell-1}\tilde{J}$ to $J_{\tilde{\mu}}$), and whatever follows the first step of the lookahead is preparation for the Newton step.

Finally, it is worth mentioning that the approximation in value space algorithm computes $J_{\tilde{\mu}}$ differently than both the PI method and the classical form of Newton's method. It does not explicitly compute any values of $J_{\tilde{\mu}}$; instead, the control is applied to the system and the cost is accumulated accordingly. Thus the values of $J_{\tilde{\mu}}$ are implicitly computed only for those x that are encountered in the system trajectory that is generated on-line.

Certainty Equivalent Approximations and the Newton Step

We noted earlier that for stochastic DP problems, ℓ-step lookahead can be computationally expensive, because the lookahead graph expands fast as ℓ increases, due to the stochastic character of the problem. The *certainty equivalence approach* is an important approximation idea for dealing with this difficulty. In the classical form of this approach, some or all of the stochastic disturbances w_k are replaced by some deterministic quantities, such as their expected values. Then a policy is computed off-line for the resulting deterministic problem, and it is used on-line for the actual stochastic problem.

The certainty equivalence approach can also be used to expedite the computations of the ℓ-step lookahead minimization. One way to do this is to simply replace each of the uncertain ℓ quantities $w_k, w_{k+1}, \ldots, w_{k+\ell-1}$ by a deterministic value $\overline{w}$. Conceptually, this replaces the Bellman operators T and T_μ,

$$(TJ)(x) = \min_{u \in U(x)} E\Big\{ g(x,u,w) + \alpha J\big(f(x,u,w)\big) \Big\},$$

sive over-relaxation"); see the book by Ortega and Rheinboldt [OrR70] (Section 13.4). Their convergence rate is superlinear, similar to Newton's method, as long as they involve a pure Newton step, along with the first-order steps.

$$(T_\mu J)(x) = E\Big\{g\big(x,\mu(x),w\big) + \alpha J\big(f(x,\mu(x),w)\big)\Big\},$$

[cf. Eqs. (3.1) and (3.2)] with deterministic operators $\overline{T}$ and $\overline{T}_\mu$, given by

$$(\overline{T}J)(x) = \min_{u\in U(x)} \Big[g(x,u,\overline{w}) + \alpha J\big(f(x,u,\overline{w})\big)\Big],$$

$$(\overline{T}_\mu J)(x) = g\big(x,\mu(x),\overline{w}\big) + \alpha J\big(f(x,\mu(x),\overline{w})\big).$$

The resulting ℓ-step lookahead minimization then becomes simpler; for example, in the case of a finite control space problem, it is a deterministic shortest path computation, involving an acyclic ℓ-stage graph that expands at each stage by a factor n, where n is the size of the control space. However, this approach yields a policy $\overline{\mu}$ such that

$$\overline{T}_{\overline{\mu}}(\overline{T}^{\ell-1}\tilde{J}) = \overline{T}(\overline{T}^{\ell-1}\tilde{J}),$$

and the cost function $J_{\overline{\mu}}$ of this policy is generated by a Newton step, which aims to find a fixed point of $\overline{T}$ (not T), starting from $\overline{T}^{\ell-1}\tilde{J}$. Thus the Newton step now aims at a fixed point of $\overline{T}$, which is not equal to J^*. As a result the benefit of the Newton step is lost to a great extent.

Still, we may largely correct this difficulty, while retaining substantial simplification, by using certainty equivalence for *only the last* $\ell-1$ *stages* of the ℓ-step lookahead. This can be done with an ℓ-step lookahead scheme whereby only the uncertain quantities $w_{k+1},\ldots,w_{k+\ell-1}$ are replaced by a deterministic value $\overline{w}$, while w_k is treated as a stochastic quantity, as first proposed in the paper by Bertsekas and Castañon [BeC99]. In this way we obtain a policy $\overline{\mu}$ such that

$$T_{\overline{\mu}}(\overline{T}^{\ell-1}\tilde{J}) = T(\overline{T}^{\ell-1}\tilde{J}).$$

The cost function $J_{\overline{\mu}}$ of this policy is then generated by a Newton step, which aims to find a fixed point of T (not $\overline{T}$), starting again from $\overline{T}^{\ell-1}\tilde{J}$. Thus the benefit of the fast convergence of Newton's method is restored. In fact based on insights derived from this Newton step interpretation, it appears that the performance penalty for making the last $\ell-1$ stages of the ℓ-step lookahead deterministic is minimal when $\overline{T}^{\ell-1}\tilde{J}$ is "near" J^*. At the same time the ℓ-step minimization $T(\overline{T}^{\ell-1}\tilde{J})$ involves only one stochastic step, the first one, and hence potentially a much "thinner" lookahead graph, than the one corresponding to the ℓ-step minimization $T^\ell\tilde{J}$, which does not involve any certainty equivalence-type approximations.

The preceding discussion also points to a more general approximation idea for dealing with the onerous computational requirements of long multistep lookahead minimization. We may approximate the tail $(\ell-1)$-step portion $T^{\ell-1}\tilde{J}$ of the ℓ-step lookahead minimization with any simplified

calculation that produces an approximation $\hat{J} \approx T^{\ell-1}\tilde{J}$, and then obtain the lookahead policy $\tilde{\mu}$ using the minimization

$$T_{\tilde{\mu}}\hat{J} = T\hat{J}.$$

This type of simplification will still involve a Newton step (from $\hat{J}$ to $J_{\tilde{\mu}}$), and benefit from the corresponding fast convergence property.

Local and Global Performance Estimates Compared

The preceding Newton step interpretation of the move from $\tilde{J}$ (the terminal cost function approximation) to $J_{\tilde{\mu}}$ (the cost function of the lookahead policy $\tilde{\mu}$) suggests a superlinear performance estimate

$$\max_x \big|J_{\tilde{\mu}}(x) - J^*(x)\big| = o\Big(\max_x \big|\tilde{J}(x) - J^*(x)\big|\Big).$$

However, this estimate is *local* in character. It is meaningful only when $\tilde{J}$ is "close" to J^*. When $\tilde{J}$ is far from J^*, the difference $\max_x \big|J_{\tilde{\mu}}(x) - J^*(x)\big|$ may be large and even infinite when $\tilde{\mu}$ is unstable (see the discussion in the next section).

There are global estimates for the difference

$$\max_x \big|J_{\tilde{\mu}}(x) - J^*(x)\big|$$

for several types of problems, including the upper bound

$$\max_x \big|J_{\tilde{\mu}}(x) - J^*(x)\big| \leq \frac{2\alpha^\ell}{1-\alpha} \max_x \big|\tilde{J}(x) - J^*(x)\big|$$

for ℓ-step lookahead, and α-discounted problems where all the Bellman operators T_μ are contraction mappings; see the neurodynamic programming book [BeT96] (Section 6.1, Prop. 6.1), or the RL book [Ber20a] (Section 5.4, Prop. 5.4.1). These books also contain other related global estimates, which hold for all $\tilde{J}$, both close and far from J^*. However, these global estimates tend to be overly conservative and not representative of the performance of approximation in value space schemes when $\tilde{J}$ is near J^*. For example, for finite spaces α-discounted MDP, $\tilde{\mu}$ can be shown to be optimal when $\max_x \big|\tilde{J}(x) - J^*(x)\big|$ is sufficiently small; this can also be seen from the fact that the components $(TJ)(x)$ of the Bellman operator are not only concave but also piecewise linear, so Newton's method converges finitely. For a further comparative discussion of local and global error bounds, we refer to Appendix A.

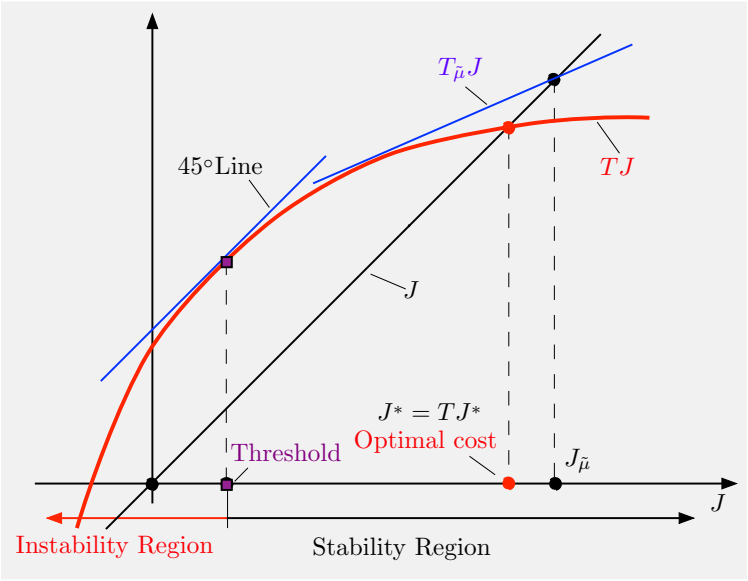

Figure 3.3.1 Illustration of the regions of stability and instability for approximation in value space with one-step lookahead. The stability region is the set of all $\tilde{J}$ such that the policy $\tilde{\mu}$ obtained from the one-step lookahead minimization $T_{\tilde{\mu}}\tilde{J} = T\tilde{J}$ satisfies $J_{\tilde{\mu}}(x) < \infty$ for all x.

3.3 REGION OF STABILITY

For any control system design method, the stability of the policy obtained is of paramount importance. It is thus essential to investigate and verify the stability of controllers obtained through approximation in value space schemes. Historically, there have been several proposed definitions of stability in control theory. Within the context of this work, our focus on stability issues will be for problems with a termination state t, which is cost-free, and with a cost per stage that is positive outside the termination state, such as the undiscounted positive cost deterministic problem introduced earlier (cf. Section 2.1). Moreover, it is best for our purposes to adopt an optimization-based definition. In particular, we say that a policy μ is *unstable* if $J_\mu(x) = \infty$ for some states x. Equivalently, we say that the policy μ *stable* if $J_\mu(x) < \infty$ for all states x. This definition has the advantage that it applies to general state and control spaces. Naturally, it can be made more specific in particular problem instances.†

† For the undiscounted positive cost deterministic problem introduced earlier (cf. Section 2.1), it can be shown that if a policy μ is stable, then J_μ is the "smallest" solution of the Bellman equation $J = T_\mu J$ within the class of nonnegative real-valued functions, and under mild assumptions it is the unique solution of

In the context of approximation in value space we are interested in the *region of stability*, which is the set of cost function approximations $\tilde{J} \in R(X)$ for which the corresponding one-step or multistep lookahead policies $\tilde{\mu}$ are stable. For discounted problems with bounded cost per stage, all policies have real-valued cost functions, so questions of stability do not arise. In general, however, the region of stability may be a strict subset of the set of real-valued functions; this will be illustrated later for the undiscounted deterministic case of the linear quadratic problem of Section 2.1 (cf. Example 2.1.1). Figure 3.3.1 illustrates the region of stability for approximation in value space with one-step lookahead.

An interesting observation from Fig. 3.3.1 is that if $\tilde{J}$ does not belong to the region of stability and $\tilde{\mu}$ is a corresponding one-step lookahead unstable policy, the Bellman equation $J = T_{\tilde{\mu}}J$ may have real-valued solutions. However, these solutions will not be equal to $J_{\tilde{\mu}}$, as this would violate the definition of region of stability. Generally, if T_μ is not a contraction mapping, T_μ may have real-valued fixed points, none of which is equal to J_μ.

Figure 3.3.2 illustrates the region of stability for the case of multistep lookahead minimization. The insights from this figure are similar to the one-step lookahead case of Fig. 3.3.1. However, the figure indicates that *the region of stability of the ℓ-step lookahead controller $\tilde{\mu}$ depends on ℓ, and tends to become larger as ℓ increases*. The reason is that ℓ-step lookahead with terminal cost $\tilde{J}$ is equivalent to one-step lookahead with terminal cost $T^{\ell-1}\tilde{J}$, which tends to be closer to the optimal cost function J^* than $\tilde{J}$ (assuming convergence of the VI method).

How Can We Obtain Function Approximations $\tilde{J}$ Within the Region of Stability?

Naturally, identifying and obtaining cost function approximations $\tilde{J}$ that lie within the region of stability with either one-step or multistep lookahead is very important within our context. We will focus on this question for the special case where the expected cost per stage is nonnegative

$$E\big\{g(x,u,w)\big\} \geq 0, \qquad \text{for all } x,\ u \in U(x),$$

and assume that J^* is real-valued. This is the case of most interest in model predictive control, but also arises in other problems of interest, including stochastic shortest path problems that involve a termination state.

From Fig. 3.3.2 it can be conjectured that if the sequence $\{T^k\tilde{J}\}$ generated by the VI algorithm converges to J^* for all $\tilde{J}$ such that $0 \leq \tilde{J} \leq J^*$

$J = T_\mu J$ within the class of nonnegative real-valued functions J with $J(t) = 0$; see the author's paper [Ber17b]. Moreover, if μ is unstable, then the Bellman equation $J = T_\mu J$ has no solution within the class of nonnegative real-valued functions.

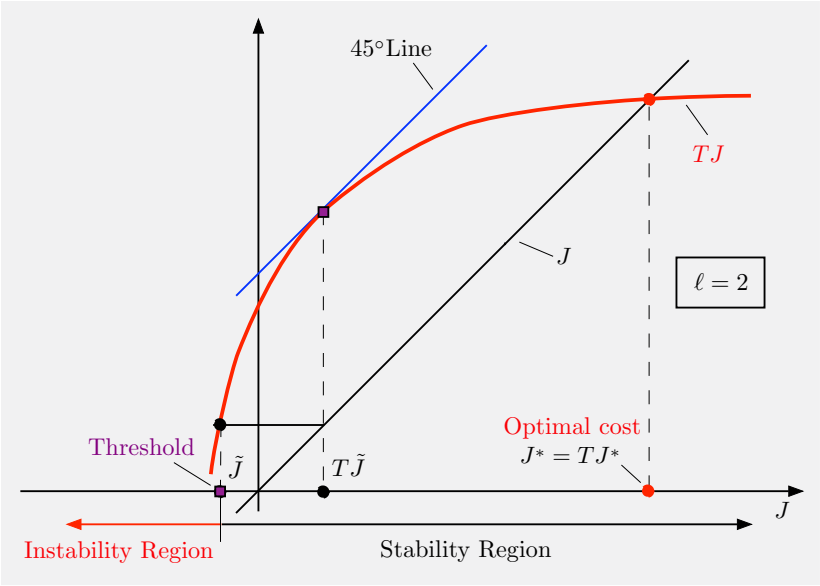

Figure 3.3.2 Illustration of the regions of stability and instability for approximation in value space with ℓ-step lookahead minimization. The stability region is the set of all $\tilde{J}$ for which the policy $\tilde{\mu}$ such that $T^\ell \tilde{J} = T_{\tilde{\mu}} T^{\ell-1} \tilde{J}$ satisfies $J_{\tilde{\mu}}(x) < \infty$ for all x (the figure shows the case $\ell = 2$). The region of instability tends to be reduced as ℓ increases.

(which is true under very general conditions; see [Ber12], [Ber22a]), then $T^{\ell-1}\tilde{J}$ belongs to the region of stability for sufficiently large ℓ. Related ideas have been discussed in the adaptive DP literature by Liu and his collaborators [HWL21], [LXZ21], [WLL16], and by Heydari [Hey17], [Hey18], who provide extensive references; see also Winnicki et al. [WLL21]. We will revisit this issue in the context of linear quadratic problems. This conjecture is generally true, but requires that, in addition to J^*, all functions $\tilde{J}$ within a neighborhood of J^* belong to the region of stability. Our subsequent discussion will aim to address this difficulty.

An important fact for our nonnegative cost problem context is that *the region of stability includes all real-valued nonnegative functions $\tilde{J}$ such that*

$$T\tilde{J} \leq \tilde{J}. \tag{3.10}$$

Indeed if $\tilde{\mu}$ is the corresponding one-step lookahead policy, we have

$$T_{\tilde{\mu}}\tilde{J} = T\tilde{J} \leq \tilde{J},$$

and from a well-known result on nonnegative cost infinite horizon problems [see [Ber12], Prop. 4.1.4(a)], it follows that

$$J_{\tilde{\mu}} \leq \tilde{J};$$

(the proof argument is that if $T_{\tilde{\mu}}\tilde{J} \leq \tilde{J}$ then $T_{\tilde{\mu}}^{k+1}\tilde{J} \leq T_{\tilde{\mu}}^{k}\tilde{J}$ for all k, so, using also the fact $0 \leq \tilde{J}$, the limit of $T_{\tilde{\mu}}^{k}\tilde{J}$, call it J_∞, satisfies $J_{\tilde{\mu}} \leq J_\infty \leq \tilde{J}$). Thus if $\tilde{J}$ is nonnegative and real-valued, $J_{\tilde{\mu}}$ is also real-valued, so $\tilde{\mu}$ is stable. It follows that $\tilde{J}$ belongs to the region of stability. This is a known result in specific contexts, such as MPC (see the book by Rawlings, Mayne, and Diehl [RMD17], Section 2.4, which contains extensive references to prior work on stability issues).

An important special case where the condition $T\tilde{J} \leq \tilde{J}$ is satisfied is when $\tilde{J}$ is the cost function of a stable policy, i.e., $\tilde{J} = J_\mu$. Then we have that J_μ is real-valued and satisfies $T_\mu J_\mu = J_\mu$, so it follows that $TJ_\mu \leq J_\mu$. This case relates to the rollout algorithm and shows that *rollout with a stable policy yields a stable lookahead policy*. It also suggests that if μ is stable, then $T_\mu^m \tilde{J}$ belongs to the region of stability for sufficiently large m.

Besides J_μ, with stable μ, and J^*, there are other interesting functions $\tilde{J}$ satisfying the stability condition $T\tilde{J} \leq \tilde{J}$. In particular, let β be a scalar with $\beta > 1$, and for a stable policy μ, consider the *β-amplified operator* $T_{\mu,\beta}$ defined by

$$(T_{\mu,\beta}J)(x) = E\Big\{\beta g\big(x,\mu(x),w\big) + \alpha J\big(f(x,\mu(x),w)\big)\Big\}, \qquad \text{for all } x.$$

Then it can be seen that the function

$$J_{\mu,\beta} = \beta J_\mu$$

is a fixed point of $T_{\mu,\beta}$ and satisfies $TJ_{\mu,\beta} \leq J_{\mu,\beta}$. This follows by writing

$$J_{\mu,\beta} = T_{\mu,\beta}J_{\mu,\beta} \geq T_\mu J_{\mu,\beta} \geq TJ_{\mu,\beta}. \tag{3.11}$$

Thus $J_{\mu,\beta}$ lies within the region of stability, and lies "further to the right" of J_μ. Thus we may conjecture that it can be more reliably approximated by $T_{\mu,\beta}^m \tilde{J}$ than J_μ is approximated by $T_\mu^m \tilde{J}$ in the context of m-step truncated rollout.

To illustrate this fact, consider a stable policy μ, and assume that the expected cost per stage at states other than a termination state t (if one exists) is bounded away from 0, i.e.,

$$C = \min_{x \neq t} E\Big\{g\big(x,\mu(x),w\big)\Big\} > 0.$$

Then we claim that given a scalar $\beta > 1$, any function $\hat{J} \in R(X)$ with $\hat{J}(t) = 0$, that satisfies

$$\max_x \big|\hat{J}(x) - J_{\mu,\beta}(x)\big| \leq \delta, \qquad \text{for all } x, \tag{3.12}$$

where

$$\delta = \frac{(\beta-1)C}{1+\alpha},$$

also satisfies the stability condition $T\hat{J} \leq \hat{J}$. From this it follows that *for a given nonnegative and real-valued $\tilde{J}$, and for sufficiently large m, so that the function $\hat{J} = T^m_{\mu,\beta}\tilde{J}$ satisfies Eq. (3.12), we have that $\hat{J}$ lies within the region of stability.*

To see this, note that for all $x \neq t$, we have

$$J_{\mu,\beta}(x) = \beta E\Big\{g\big(x,\mu(x),w\big)\Big\} + \alpha E\Big\{J_{\mu,\beta}\big(f(x,\mu(x),w)\big)\Big\},$$

so that by using Eq. (3.12), we have

$$\hat{J}(x) + \delta \geq \beta E\Big\{g\big(x,\mu(x),w\big)\Big\} + \alpha E\Big\{\hat{J}\big(f(x,\mu(x),w)\big)\Big\} - \alpha\delta.$$

It follows that

$$\begin{aligned}
\hat{J}(x) &\geq E\Big\{g\big(x,\mu(x),w\big)\Big\} + \alpha E\Big\{\hat{J}\big(f(x,\mu(x),w)\big)\Big\} \\
&\quad + (\beta-1)E\Big\{g\big(x,\mu(x),w\big)\Big\} - (1+\alpha)\delta \\
&\geq E\Big\{g\big(x,\mu(x),w\big)\Big\} + \alpha E\Big\{\hat{J}\big(f(x,\mu(x),w)\big)\Big\} + (\beta-1)C - (1+\alpha)\delta \\
&= (T_\mu \hat{J})(x) \\
&\geq (T\hat{J})(x),
\end{aligned}$$

so the stability condition $T\hat{J} \leq \hat{J}$ is satisfied.

Similarly the function

$$J^*_\beta = \beta J^*$$

is a fixed point of the operator T_β defined by

$$(T_\beta J)(x) = \min_{u\in U(x)} E\Big\{\beta g(x,u,w) + \alpha J\big(f(x,u,w)\big)\Big\}, \qquad \text{for all } x.$$

It can be seen, using an argument similar to Eq. (3.11), that J^*_β satisfies $TJ^*_\beta \leq J^*_\beta$, so it lies within the region of stability. Furthermore, similar to the case of truncated rollout discussed earlier, we may conjecture that J^*_β can be more reliably approximated by $T^{\ell-1}_\beta \tilde{J}$ than J^* is approximated by $T^{\ell-1}\tilde{J}$ in the context of ℓ-step lookahead.

3.4 POLICY ITERATION, ROLLOUT, AND NEWTON'S METHOD

Another major class of infinite horizon algorithms is based on *policy iteration* (PI for short), which involves the repeated use of policy improvement, in analogy with the AlphaZero/TD-Gammon off-line training algorithms,

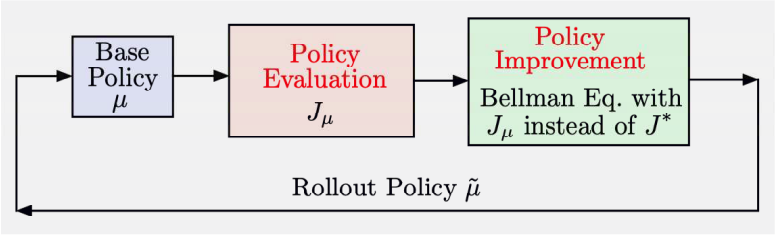

Figure 3.4.1 Schematic illustration of PI as repeated rollout. It generates a sequence of policies, with each policy μ in the sequence being the base policy that generates the next policy $\tilde{\mu}$ in the sequence as the corresponding rollout policy.

described in Chapter 1. Each iteration of the PI algorithm starts with a stable policy (which we call *current* or *base* policy), and generates another stable policy (which we call *new* or *rollout* policy, respectively). For the infinite horizon problem of Section 2.1, given the base policy μ, the iteration consists of two phases (see Fig. 3.4.1):

(a) *Policy evaluation*, which computes the cost function J_μ. One possibility is to solve the corresponding Bellman equation

$$J_\mu(x) = E\Big\{g\big(x,\mu(x),w\big) + \alpha J_\mu\big(f(x,\mu(x),w)\big)\Big\}, \qquad \text{for all } x. \tag{3.13}$$

However, the value $J_\mu(x)$ for any x can also be computed by Monte Carlo simulation, by averaging over many randomly generated trajectories the cost of the policy starting from x. Other, more sophisticated possibilities include the use of specialized simulation-based methods, such as *temporal difference methods*, for which there is extensive literature (see e.g., the books [BeT96], [SuB98], [Ber12]).

(b) *Policy improvement*, which computes the rollout policy $\tilde{\mu}$ using the one-step lookahead minimization

$$\tilde{\mu}(x) \in \arg\min_{u\in U(x)} E\Big\{g(x,u,w) + \alpha J_\mu\big(f(x,u,w)\big)\Big\}, \qquad \text{for all } x. \tag{3.14}$$

It is generally expected (and can be proved under mild conditions) that the rollout policy is improved in the sense that

$$J_{\tilde{\mu}}(x) \leq J_\mu(x), \qquad \text{for all } x.$$

Proofs of this fact in a variety of contexts can be found in most DP books, including the author's [Ber12], [Ber18a], [Ber19a], [Ber20a], [Ber22a].

Thus PI generates a sequence of stable policies $\{\mu^k\}$, by obtaining μ^{k+1} through a policy improvement operation using J_{μ^k} in place of J_μ

in Eq. (3.14), which is obtained through policy evaluation of the preceding policy μ^k using Eq. (3.13). It is well known that (exact) PI has solid convergence properties; see the DP textbooks cited earlier, as well as the author's RL book [Ber19a]. These properties hold even when the method is implemented (with appropriate modifications) in unconventional computing environments, involving asynchronous distributed computation, as shown in a series of papers by Bertsekas and Yu [BeY10], [BeY12], [YuB13].

In terms of our abstract notation, the PI algorithm can be written in a compact form. For the generated policy sequence $\{\mu^k\}$, the policy evaluation phase obtains J_{μ^k} from the equation

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}, \tag{3.15}$$

while the policy improvement phase obtains μ^{k+1} through the equation

$$T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}. \tag{3.16}$$

As Fig. 3.4.2 illustrates, PI can be viewed as Newton's method for solving the Bellman equation in the function space of cost functions J. In particular, *the policy improvement Eq. (3.16) is the Newton step starting from J_{μ^k}, and yields μ^{k+1} as the corresponding one-step lookahead/rollout policy.* Figure 3.4.3 illustrates the rollout algorithm, which is just the first iteration of PI.

In contrast to approximation in value space, the interpretation of PI in terms of Newton's method has a long history. We refer to the original works for linear quadratic problems by Kleinman [Klei68],† and for finite-state infinite horizon discounted and Markov game problems by Pollatschek and Avi-Itzhak [PoA69] (who also showed that the method may oscillate in the game case). Subsequent works, which discuss algorithmic variations and approximations, include Hewer [Hew71], Puterman and Brumelle [PuB78], [PuB79], Santos and Rust [SaR04], Bokanowski, Maroso, and Zidani [BMZ09], Hylla [Hyl11], Magirou, Vassalos, and Barakitis [MVB20], Bertsekas [Ber21c], and Kundu and Kunitsch [KuK21]. Some of these papers address broader classes of problems (such as continuous-time optimal control, minimax problems, and Markov games), and include superlinear convergence rate results under various (often restrictive) assumptions, as well as PI variants. Early related works for control system design include Saridis and Lee [SaL79], Beard [Bea95], and Beard, Saridis, and Wen [BSW99].

† This was part of Kleinman's Ph.D. thesis [Kle67] at M.I.T., supervised by M. Athans. Kleinman gives credit for the one-dimensional version of his results to Bellman and Kalaba [BeK65]. Note also that the first proposal of the PI method was given by Bellman in his classic book [Bel57], under the name "approximation in policy space."

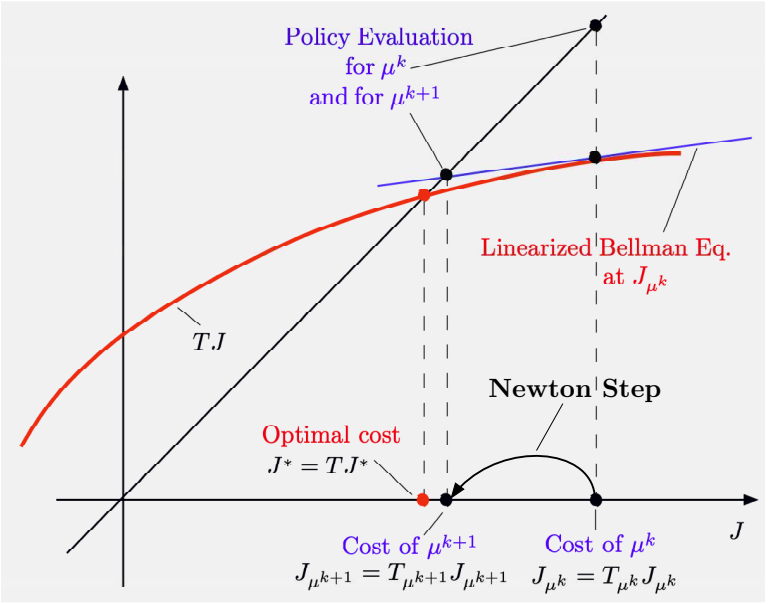

Figure 3.4.2 Geometric interpretation of a policy iteration. Starting from the stable current policy μ^k, it evaluates the corresponding cost function J_{μ^k}, and computes the next policy μ^{k+1} according to

$$T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}.$$

The corresponding cost function $J_{\mu^{k+1}}$ is obtained as the solution of the linearized equation $J = T_{\mu^{k+1}} J$, so it is the result of a Newton step for solving the Bellman equation $J = TJ$, starting from J_{μ^k}. Note that in policy iteration, the Newton step always starts at a function J_μ, which satisfies $J_\mu \geq J^*$ as well as $TJ_\mu \leq J_\mu$ (cf. our discussion on stability in Section 3.3).

Rollout

Generally, rollout with a stable base policy μ can be viewed as a single iteration of Newton's method starting from J_μ, as applied to the solution of the Bellman equation (see Fig. 3.4.3). Note that rollout/policy improvement is applied just at the current state during real-time operation of the system. This makes the on-line implementation possible, even for problems with very large state space, provided that the policy evaluation of the base policy can be done on-line as needed. For this we often need on-line deterministic or stochastic simulation from each of the states x_k generated by the system in real time.

As Fig. 3.4.3 illustrates, the cost function of the rollout policy $J_{\tilde{\mu}}$ is obtained by constructing a linearized version of Bellman's equation at J_μ (its linear approximation at J_μ), and then solving it. If the function TJ is nearly linear (i.e., has small "curvature") the rollout policy performance

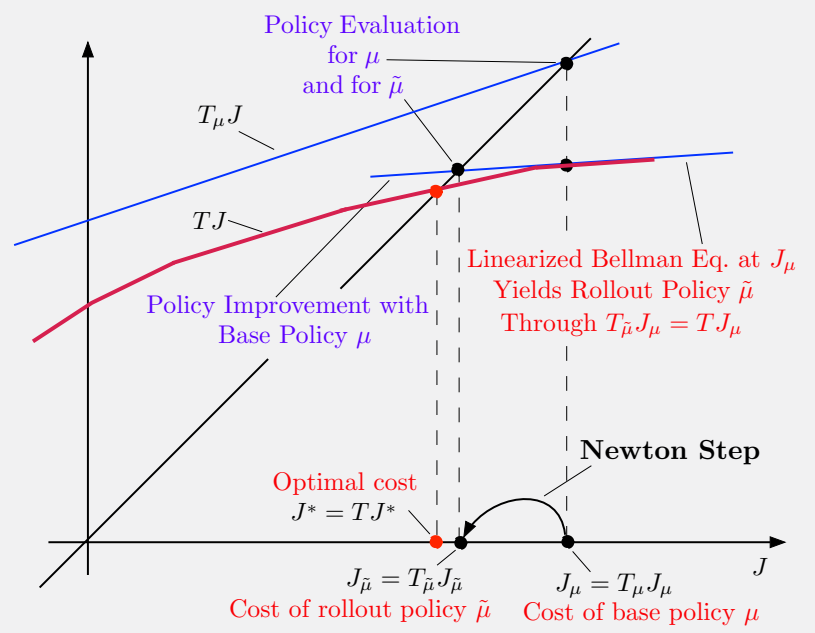

Figure 3.4.3 Geometric interpretation of rollout. Each policy μ defines the linear function $T_\mu J$ of J, given by Eq. (3.2), and TJ is the function given by Eq. (3.1), which can also be written as $TJ = \min_\mu T_\mu J$. The figure shows a policy iteration starting from a base policy μ. It computes J_μ by policy evaluation (by solving the linear equation $J = T_\mu J$ as shown). It then performs a policy improvement using μ as the base policy to produce the rollout policy $\tilde{\mu}$ as shown: the cost function of the rollout policy, $J_{\tilde{\mu}}$, is obtained by solving the version of Bellman's equation that is linearized at the point J_μ, as in Newton's method.

$J_{\tilde{\mu}}(x)$ is very close to the optimal $J^*(x)$, even if the base policy μ is far from optimal. This explains the large cost improvements that are typically observed in practice with the rollout algorithm.

An interesting question is how to compare the rollout performance $J_{\tilde{\mu}}(x)$ for a given initial state x, with the base policy performance $J_\mu(x)$. Clearly, we would like $J_\mu(x) - J_{\tilde{\mu}}(x)$ to be large, but this is not the right way to look at cost improvement. The reason is that $J_\mu(x) - J_{\tilde{\mu}}(x)$ will be small if its upper bound, $J_\mu(x) - J^*(x)$, is small, i.e., if the base policy is close to optimal. What is important is that the error ratio

$$\frac{J_{\tilde{\mu}}(x) - J^*(x)}{J_\mu(x) - J^*(x)} \tag{3.17}$$

is small. Indeed, this ratio becomes smaller as $J_\mu(x) - J^*(x)$ approaches 0 because of the superlinear convergence rate of Newton's method that underlies the rollout algorithm (cf. Fig. 3.4.3). Unfortunately, it is hard to evaluate this ratio, since we do not know $J^*(x)$. On the other hand, we

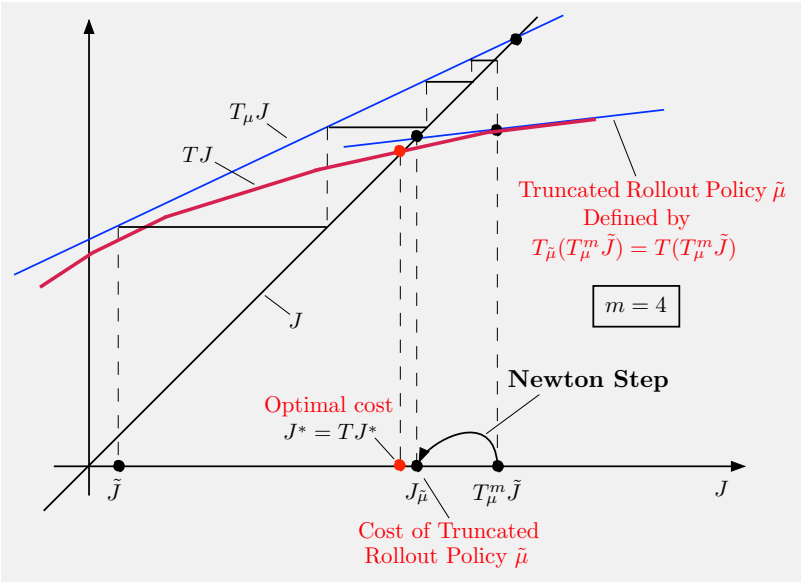

Figure 3.4.4 Geometric interpretation of truncated rollout with one-step lookahead minimization, m value iterations with the base policy μ, and a terminal cost function approximation $\tilde{J}$ (here $m = 4$).

should not be underwhelmed if we observe a small performance improvement $J_\mu(x) - J_{\tilde{\mu}}(x)$: the reason may be that the base policy is already near-optimal, and in fact we are likely doing very well in terms of the ratio (3.17).

Truncated Rollout and Optimistic Policy Iteration

Variants of rollout may involve multistep lookahead, truncation, and terminal cost function approximation, as in the case of AlphaZero/TD-Gammon, cf. Chapter 1. These variants admit geometric interpretations that are similar to the ones given earlier; see Fig. 3.4.4. Truncated rollout uses m VIs with the base policy μ and a terminal cost function approximation $\tilde{J}$ to approximate the cost function J_μ.

In the case of one-step lookahead, the truncated rollout policy $\tilde{\mu}$ is defined by

$$T_{\tilde{\mu}}(T_\mu^m \tilde{J}) = T(T_\mu^m \tilde{J}), \tag{3.18}$$

i.e., $\tilde{\mu}$ attains the minimum when the Bellman operator T is applied to the function $T_\mu^m \tilde{J}$ (the cost obtained by using the base policy μ for m steps followed by terminal cost approximation $\tilde{J}$); see Fig. 3.4.4. In the case of ℓ-step lookahead, the truncated rollout policy $\tilde{\mu}$ is defined by

$$T_{\tilde{\mu}}(T^{\ell-1} T_\mu^m \tilde{J}) = T(T^{\ell-1} T_\mu^m \tilde{J}). \tag{3.19}$$

Truncated rollout is related to a variant of PI called *optimistic*. This variant approximates the policy evaluation step by using m value iterations using the base policy μ; see [BeT96], [Ber12], [Ber19a] for a more detailed discussion of this relation. A method that is related to optimistic PI is the λ-PI method, which is related to the proximal algorithm of convex analysis, and is discussed in several of the author's books ([BeT96], [Ber12], [Ber20a], [Ber22a]), and papers (BeI96], [Ber15], [Ber18d]), and can also be used to define the one-step lookahead policy in place of Eq. (3.18). In particular, Section 6 of the paper [Ber18d] is focused on λ-PI methods, which serve as approximations to the ordinary PI/Newton methods for finite-state discounted and SSP problems.

As noted earlier, variants of Newton's method that involve multiple fixed point iterations, before and after each Newton step, but without truncated rollout, i.e.,

$$T_{\tilde{\mu}}(T^{\ell-1}\tilde{J}) = T(T^{\ell-1}\tilde{J}), \tag{3.20}$$

are well-known. The classical numerical analysis book by Ortega and Rheinboldt [OrR70] (Sections 13.3 and 13.4) provides various convergence results, under assumptions that include differentiability and convexity of the components of T, and nonnegativity of the inverse Jacobian of T. These assumptions, particularly differentiability, may not be satisfied within our DP context. Moreover, for methods of the form (3.20), the initial point must satisfy an additional assumption, which ensures that the convergence to J^* is monotonic from above (in this case, if in addition the Jacobian of T is isotone, an auxiliary sequence can be constructed that converges to J^* monotonically from below; see [OrR70], 13.3.4, 13.4.2). This is similar to existing convergence results for the optimistic PI method in DP; see e.g., [BeT96], [Ber12].

Geometrical interpretations such as the ones of Fig. 3.4.4 suggest, among others, that:

(a) The cost improvement $J_\mu - J_{\tilde{\mu}}$, from base to rollout policy, tends to become larger as the length ℓ of the lookahead increases.

(b) Truncated rollout with ℓ-step lookahead minimization, followed by m steps of a base policy μ, and then followed by terminal cost function approximation $\tilde{J}$ may be viewed, under certain conditions, as an economic alternative to $(\ell + m)$-step lookahead minimization using $\tilde{J}$ as terminal cost function approximation.

Figure 3.4.5 illustrates in summary the approximation in value space scheme with ℓ-step lookahead minimization and m-step truncated rollout [cf. Eq. (3.19)], and its connection to Newton's method. This figure indicates the parts that are ordinarily associated with on-line play and off-line training, and parallels our earlier Fig. 1.2.1, which applies to AlphaZero, TD-Gammon, and related on-line schemes.

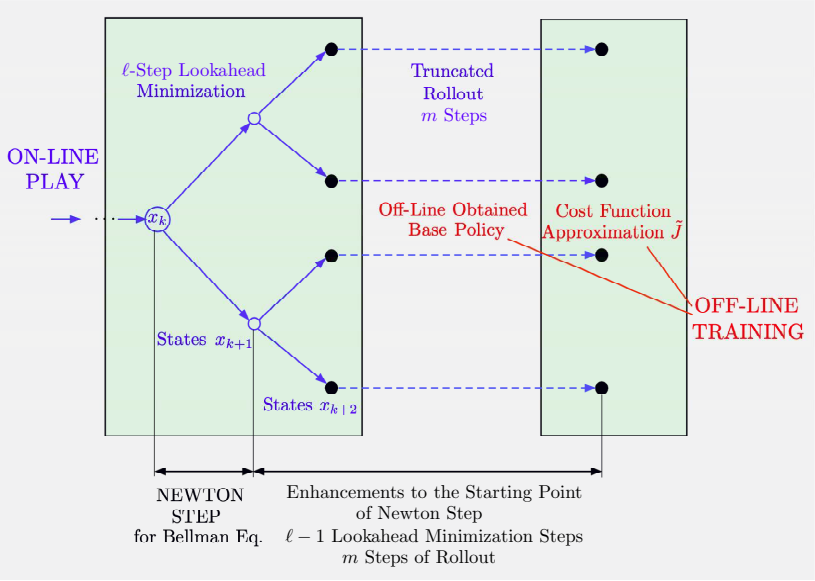

Figure 3.4.5 Illustration of the approximation in value space scheme with ℓ-step lookahead minimization ($\ell = 2$ in the figure) and m-step truncated rollout [cf. Eq. (3.19)], and its connection to Newton's method. The Newton step for solving the Bellman equation $J^* = TJ^*$ corresponds to the first (out of ℓ steps of) lookahead minimization. The remaining $\ell - 1$ steps of lookahead minimization (VI iterations), and the m truncated rollout steps (VI iterations with the base policy), improve the starting point of the Newton step, from its off-line-obtained cost function approximation $\tilde{J}$.

Lookahead Length Issues in Truncated Rollout

A question of practical interest is how to choose the lookahead lengths ℓ and m in truncated rollout schemes. It is clear that large values ℓ for lookahead minimization are beneficial (in the sense of producing improved lookahead policy cost functions $J_{\tilde{\mu}}$), since additional VI iterations bring closer to J^* the starting point $T^{\ell-1}\tilde{J}$ of the Newton step. Note, however, that while long lookahead minimization is computationally expensive (its complexity increases exponentially with ℓ), it is only the first stage of the multistep lookahead that contributes to the Newton step, while the remaining $\ell - 1$ steps are far less effective first order/VI iterations.

Regarding the value of m, long truncated rollout brings the starting point for the Newton step closer to J_μ, but not necessarily closer to J^*, as indicated by Fig. 3.4.4. Indeed computational experiments suggest that increasing values for m may be counterproductive beyond some threshold, and that this threshold generally depends on the problem and the terminal cost approximation $\tilde{J}$; see also our subsequent discussion for linear quadratic problems in Section 4.6. This is also consistent with long stand-

ing experience with optimistic policy iteration, which is closely connected with truncated rollout, as noted earlier. Unfortunately, however, there is no analysis that can illuminate this issue, and the available error bounds for truncated rollout (see [Ber19a], [Ber20a]) are conservative and provide limited guidance in this regard.

Another important fact to keep in mind is that the truncated rollout steps are much less demanding computationally than the lookahead minimization steps. Thus, with other concerns weighing equally, it is computationally preferable to use large values of m rather than large values of ℓ (this was the underlying motivation for truncated rollout in Tesauro's TD-Gammon [TeG96]). On the other hand, while large values of m may be computationally tolerable in some cases, it is possible that even relatively small values of m can be computationally prohibitive. This is especially true for stochastic problems where the width of the lookahead graph tends to grow quickly.

An interesting property, which holds in some generality, is that *truncated rollout with a stable policy has a beneficial effect on the stability properties of the lookahead policy*. The reason is that the cost function J_μ of the base policy μ lies well inside the region of stability, as noted in Section 3.2. Moreover value iterations with μ (i.e., truncated rollout) tend to push the starting point of the Newton step towards J_μ. Thus a sufficient number of these value iterations will bring the starting point of the Newton step within the region of stability.

The preceding discussion suggests the following qualitative question: *is lookahead by rollout an economic substitute for lookahead by minimization?* The answer to this seems to be a qualified yes: for a given computational budget, judiciously balancing the values of m and ℓ tends to give better lookahead policy performance than simply increasing ℓ as much as possible, while setting $m = 0$ (which corresponds to no rollout). This is consistent with intuition obtained through geometric constructions such as Fig. 3.4.4, but it is difficult to establish conclusively. We discuss this issue further in Section 4.6 for the case of linear quadratic problems.

3.5 HOW SENSITIVE IS ON-LINE PLAY TO THE OFF-LINE TRAINING PROCESS?

An important issue to consider in approximation in value space is errors in the one-step or multistep minimization, or in the choice of terminal cost approximation $\tilde{J}$. Such errors are often unavoidable because the control constraint set $U(x)$ is infinite, or because the minimization is simplified for reasons of computational expediency (see our subsequent discussion of multiagent problems). Moreover, to these errors, we may add the effect of errors due to rollout truncation, and errors due to changes in problem parameters, which are reflected in changes in Bellman's equation (see our subsequent discussion of robust and adaptive control).

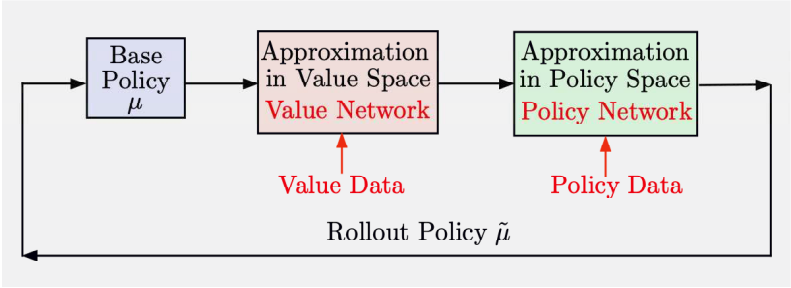

Figure 3.5.1 Schematic illustration of approximate PI. Either the policy evaluation and policy improvement phases (or both) are approximated with a value or a policy network, respectively. These could be neural networks, which are trained with (state, cost function value) data that is generated using the current base policy μ, and with (state, rollout policy control) data that is generated using the rollout policy $\tilde{\mu}$.

Note that there are three different types of approximate implementation involving: 1) a value network but no policy network (here the value network defines a policy via one-step or multistep lookahead), or 2) a policy network but no value network (here the policy network has a corresponding value function that can be computed by rollout), or 3) both a policy and a value network (the approximation architecture of AlphaZero is a case in point).

Under these circumstances the linearization of the Bellman equation at the point $\tilde{J}$ in Fig. 3.4.4 is perturbed, and the corresponding point $T_\mu^m \tilde{J}$ in Fig. 3.4.4 is also perturbed. However, the effect of these perturbations tends to be mitigated by the Newton step that produces the policy $\tilde{\mu}$ and the corresponding cost function $J_{\tilde{\mu}}$. The Newton step has a superlinear convergence property, so for an $O(\epsilon)$-order error [i.e., $O(\epsilon)/\epsilon$ stays bounded as $\epsilon \to 0$] in the calculation of $T_\mu^m \tilde{J}$, the error in $J_{\tilde{\mu}}$ will be of the much smaller order $o(\epsilon)$ [i.e., $o(\epsilon)/\epsilon \to 0$ as $\epsilon \to 0$], when $J_{\tilde{\mu}}$ is near J^*.† This is a significant insight, as it suggests that *extreme accuracy and fine-tuning of the choice of $\tilde{J}$ may not produce significant effects in the resulting performance of the one-step and particularly a multistep lookahead policy*; see also the quantitative analysis for linear quadratic problems in Section 4.5.

Approximate Policy Iteration and Implementation Errors

Both policy evaluation and policy improvement can be approximated, possibly by using training with data and approximation architectures, such as neural networks; see Fig. 3.5.1. Other approximations include simulation-based methods such as truncated rollout, and temporal difference methods

† A rigorous proof of this requires differentiability of T at $\tilde{J}$. Since T is differentiable at almost all points J, the sensitivity property just stated, will likely hold in practice even if T is not differentiable. See the appendix, which also compares global and local error bounds for approximation in value space, and for approximate PI (cf. Sections A.3 and A.4)

for policy evaluation, which involve the use of basis functions. Moreover, multistep lookahead may be used in place of one-step lookahead, and simplified minimization, based for example on multiagent rollout, may also be used. Let us also mention the possibility of a combined rollout and PI algorithm, whereby we use PI for on-line policy improvement of the base policy, by using data collected during the rollout process. This idea is relatively new and has not been tested extensively; see the subsequent discussion in Section 3.8 and the author's paper [Ber21a].

Long-standing practical experience with approximate PI is consistent with the view of the effect of implementation errors outlined above, and suggests that substantial changes in the policy evaluation and policy improvement operations often have small but largely unpredictable effects on the performance of the policies generated. For example, when TD(λ)-type methods are used for policy evaluation, the choice of λ has a large effect on the corresponding policy cost function approximations, but often has little and unpredictable effect on the performance of the generated policies through on-line play. A plausible conjecture here is that the superlinear convergence property of the exact Newton step "smooths out" the effect of off-line approximation errors.

3.6 WHY NOT JUST TRAIN A POLICY NETWORK AND USE IT WITHOUT ON-LINE PLAY?

This is a sensible and common question, which stems from the mindset that neural networks have extraordinary function approximation properties. In other words, why go through the arduous on-line process of lookahead minimization, if we can do the same thing off-line and represent the lookahead policy with a trained policy network? More generally, it is possible to use *approximation in policy space*, a major alternative approach to approximation in value space, whereby we select the policy from a suitably restricted class of policies, such as a parametric class of the form $\mu(x, r)$, where r is a parameter vector. We may then estimate r using some type of off-line training process. There are quite a few methods for performing this type of training, such as policy gradient and random search methods (see the books [SuB18] and [Ber19a] for an overview). Alternatively, some approximate DP or classical control system design method may be used.

An important advantage of approximation in policy space is that once the parametrized policy is obtained, the on-line computation of controls $\mu(x, r)$ is often much faster compared with on-line lookahead minimization. For this reason, approximation in policy space can be used to provide an approximate implementation of a known policy (no matter how obtained) for the purpose of convenient use. On the negative side, because parametrized approximations often involve substantial calculations, they are not well suited for on-line replanning.

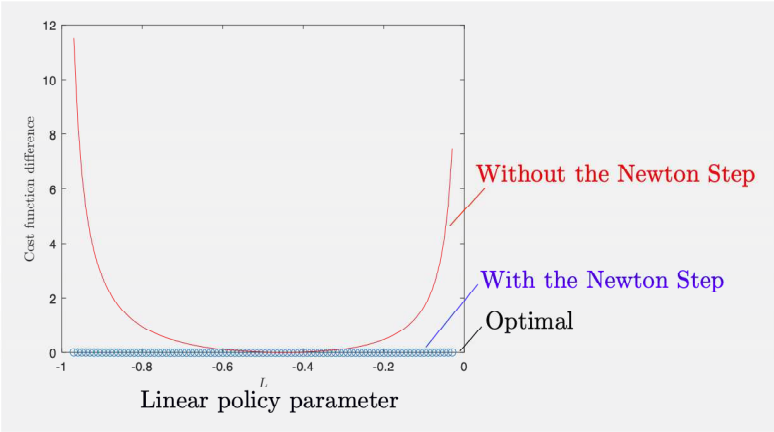

Figure 3.6.1 Illustration of the performance enhancement obtained by rollout with an off-line trained base policy for the linear quadratic problem. Here the system equation is $x_{k+1} = x_k + 2u_k$, and the cost function parameters are $q = 1$, $r = 0.5$. The optimal policy is $\mu^*(x) = L^*x$ with $L^* \approx -0.4$, and the optimal cost function is $J^*(x) = K^*x^2$, where $K^* \approx 1.1$. We consider policies of the form $\mu(x) = Lx$, where L is the parameter, with cost function of the form $J_\mu(x) = K_L x^2$. The figure shows the quadratic cost coefficient differences $K_L - K^*$ and $K_{\tilde{L}} - K^*$ as a function of L, where K_L and $K_{\tilde{L}}$ are the quadratic cost coefficients of μ (without one-step lookahead/Newton step) and the corresponding one-step lookahead policy $\tilde{\mu}$ (with one-step lookahead/Newton step).

From our point of view in this book, there is another important reason why approximation in value space is needed on top of approximation in policy space: *the off-line trained policy may not perform nearly as well as the corresponding one-step or multistep lookahead/rollout policy*, because it lacks the extra power of the associated exact Newton step (cf. our discussion of AlphaZero and TD-Gammon in Chapter 1). Figure 3.6.1 illustrates this fact with a one-dimensional linear-quadratic example, and compares the performance of a linear policy, defined by a scalar parameter, with its corresponding one-step lookahead policy.

3.7 MULTIAGENT PROBLEMS AND MULTIAGENT ROLLOUT

A major difficulty in the implementation of value space approximation with one-step lookahead is the minimization operation over $U(x)$ at a state x. When $U(x)$ is infinite, or even when it is finite but has a very large number of elements, the minimization may become prohibitively time consuming. In the case of multistep lookahead the computational difficulty becomes even more acute. In this section we discuss how to deal with this difficulty when the control u consists of m components, $u = (u_1, \ldots, u_m)$, with a separable control constraint for each component, $u_\ell \in U_\ell(x)$, $\ell = 1, \ldots, m$.

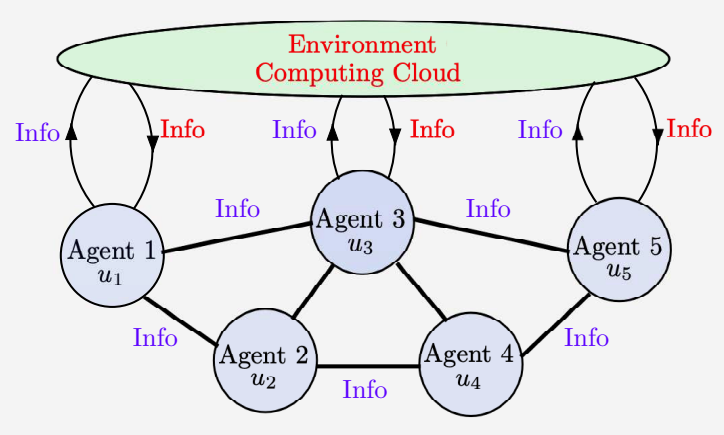

Figure 3.7.1 Schematic illustration of a multiagent problem. There are multiple "agents," and each agent $\ell = 1, \ldots, m$, controls its own decision variable u_ℓ. At each stage, agents exchange new information and also exchange information with the "environment," and then select their decision variables for the stage.

Thus the control constraint set is the Cartesian product

$$U(x) = U_1(x) \times \cdots \times U_m(x), \tag{3.21}$$

where the sets $U_\ell(x)$ are given. This structure is inspired by applications involving distributed decision making by multiple agents with communication and coordination between the agents; see Fig. 3.7.1.

To illustrate our approach, let us consider the discounted infinite horizon problem, and for the sake of the following discussion, assume that each set $U_\ell(x)$ is finite. Then the one-step lookahead minimization of the standard rollout scheme with base policy μ is given by

$$\tilde{u} \in \arg\min_{u \in U(x)} E\Big\{ g(x,u,w) + \alpha J_\mu\big(f(x,u,w)\big) \Big\}, \tag{3.22}$$

and involves as many as n^m terms, where n is the maximum number of elements of the sets $U_\ell(x)$ [so that n^m is an upper bound to the number of controls in $U(x)$, in view of its Cartesian product structure (3.21)]. Thus the standard rollout algorithm requires an exponential [order $O(n^m)$] number of computations per stage, which can be overwhelming even for moderate values of m.

This potentially large computational overhead motivates a far more computationally efficient rollout algorithm, whereby the one-step lookahead minimization (3.22) is replaced by a sequence of m successive minimizations, *one-agent-at-a-time*, with the results incorporated into the subsequent minimizations. In particular, at state x we perform the sequence of

minimizations

$$\tilde{\mu}_1(x) \in \arg\min_{u_1 \in U_1(x)} E_w\Big\{g\big(x, u_1, \mu_2(x), \ldots, \mu_m(x), w\big) + \alpha J_\mu\big(f(x, u_1, \mu_2(x), \ldots, \mu_m(x), w)\big)\Big\},$$

$$\tilde{\mu}_2(x) \in \arg\min_{u_2 \in U_2(x)} E_w\Big\{g\big(x, \tilde{\mu}_1(x), u_2, \mu_3(x) \ldots, \mu_m(x), w\big) + \alpha J_\mu\big(f(x, \tilde{\mu}_1(x), u_2, \mu_3(x), \ldots, \mu_m(x), w)\big)\Big\},$$

$$\cdots \qquad \cdots \qquad \cdots \qquad \cdots$$

$$\tilde{\mu}_m(x) \in \arg\min_{u_m \in U_m(x)} E_w\Big\{g\big(x, \tilde{\mu}_1(x), \tilde{\mu}_2(x), \ldots, \tilde{\mu}_{m-1}(x), u_m, w\big) + \alpha J_\mu\big(f(x, \tilde{\mu}_1(x), \tilde{\mu}_2(x), \ldots, \tilde{\mu}_{m-1}(x), u_m, w)\big)\Big\}.$$

Thus each agent component u_ℓ is obtained by a minimization with the preceding agent components $u_1, \ldots, u_{\ell-1}$ fixed at the previously computed values of the rollout policy, and the following agent components $u_{\ell+1}, \ldots, u_m$ fixed at the values given by the base policy. This algorithm requires order $O(nm)$ computations per stage, a potentially huge computational saving over the order $O(n^m)$ computations required by standard rollout.

A key idea here is that the computational requirements of the rollout one-step minimization (3.22) are proportional to the number of controls in the set $U(x)$ and are independent of the size of the state space. This motivates a reformulation of the problem, first suggested in the book [BeT96], Section 6.1.4, whereby control space complexity is traded off with state space complexity, by "unfolding" the control u_k into its m components, which are applied *one-agent-at-a-time* rather than all-agents-at-once.

In particular, we can reformulate the problem by breaking down the collective decision u_k into m sequential component decisions, thereby reducing the complexity of the control space while increasing the complexity of the state space. The potential advantage is that the extra state space complexity does not affect the computational requirements of some RL algorithms, including rollout.

To this end, we introduce a modified but equivalent problem, involving one-at-a-time agent control selection. At the generic state x, we break down the control u into the sequence of the m controls $u_1, u_2, \ldots, u_m$, and between x and the next state $\bar{x} = f(x, u, w)$, we introduce artificial intermediate "states" $(x, u_1), (x, u_1, u_2), \ldots, (x, u_1, \ldots, u_{m-1})$, and corresponding transitions. The choice of the last control component u_m at "state" $(x, u_1, \ldots, u_{m-1})$ marks the transition to the next state $\bar{x} = f(x, u, w)$ according to the system equation, while incurring cost $g(x, u, w)$; see Fig. 3.7.2.

It is evident that this reformulated problem is equivalent to the original, since any control choice that is possible in one problem is also possible

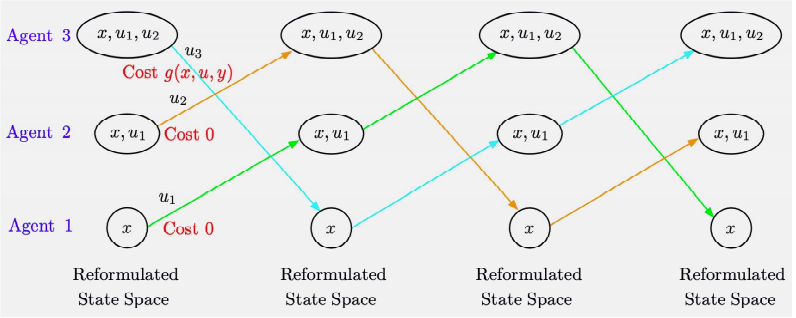

Figure 3.7.2 Equivalent formulation of the stochastic optimal control problem for the case where the control u consists of m components $u_1, u_2, \ldots, u_m$:

$$u = (u_1, \ldots, u_m) \in U_1(x_k) \times \cdots \times U_m(x_k).$$

The figure depicts the kth stage transitions. Starting from state x, we generate the intermediate states

$$(x, u_1), (x_k, u_1, u_2), \ldots, (x, u_1, \ldots, u_{m-1}),$$

using the respective controls $u_1, \ldots, u_{m-1}$. The final control u_m leads from $(x, u_1, \ldots, u_{m-1})$ to $\bar{x} = f(x, u, w)$, and the random cost $g(x, u, w)$ is incurred.

in the other problem, while the cost structure of the two problems is the same. In particular, every policy $\big(\mu_1(x), \ldots, \mu_m(x)\big)$ of the original problem, is admissible for the reformulated problem, and has the same cost function for the original as well as the reformulated problem. Reversely, every policy for the reformulated problem can be converted into a policy for the original problem that produces the same state and control trajectories and has the same cost function.

The motivation for the reformulated problem is that the control space is simplified at the expense of introducing $m-1$ additional layers of states, and the corresponding $m-1$ cost functions

$$J^1(x, u_1), J^2(x, u_1, u_2), \ldots, J^{m-1}(x, u_1, \ldots, u_{m-1}).$$

The increase in size of the state space does not adversely affect the operation of rollout, since the minimization (3.22) is performed for just one state at each stage.

A major fact that follows from the preceding reformulation is that despite the dramatic reduction in computational cost, *multiagent rollout still achieves cost improvement*:

$$J_{\tilde{\mu}}(x) \leq J_{\mu}(x), \qquad \text{for all } x,$$

where $J_{\mu}(x)$ is the cost function of the base policy $\mu = (\mu_1, \ldots, \mu_m)$, and $J_{\tilde{\mu}}(x)$ is the cost function of the rollout policy $\tilde{\mu} = (\tilde{\mu}_1, \ldots, \tilde{\mu}_m)$, starting

from state x. Furthermore, this cost improvement property can be extended to multiagent PI schemes that involve one-agent-at-a-time policy improvement operations, and have sound convergence properties (see the book [Ber20a], Chapters 3 and 5, as well as the author's papers [Ber19b], [Ber19c], [Ber20b], [Ber21b], and the paper by Bhatacharya et al. [BKB20]).

Another fact that follows from the preceding reformulation is that *multiagent rollout may be viewed as a Newton step applied to the Bellman equation that corresponds to the reformulated problem*, with starting point J_μ. This is very important for our purposes in the context of this book. In particular, *the superlinear cost improvement of the Newton step can still be obtained through multiagent rollout*, even though the amount of computation for the lookahead minimization is dramatically reduced through one-agent-at-a-time minimization. This explains experimental results given in the paper [BKB20], which have shown comparable performance for multiagent and standard rollout in the context of a large-scale multi-robot POMDP application.

Let us also mention that multiagent rollout can become the starting point for various related PI schemes that are well suited for distributed operation in important practical contexts involving multiple autonomous decision makers (see the book [Ber20a], Section 5.3.4, and the papers [Ber21b]).

Multiagent Approximation in Value Space

Let us now consider the reformulated problem of Fig. 3.7.2 and see how we can apply approximation in value space with one-step lookahead minimization, truncated rollout with a base policy $\mu = (\mu_1, \ldots, \mu_m)$, and terminal cost function approximation $\tilde{J}$.

In one such scheme that involves one-agent-at-a-time minimization, at state x we perform the sequence of minimizations

$$\tilde{u}_1 \in \arg\min_{u_1 \in U_1(x)} E_w\Big\{g\big(x, u_1, \mu_2(x), \ldots, \mu_m(x), w\big) + \alpha \tilde{J}\big(f(x, u_1, \mu_2(x), \ldots, \mu_m(x), w)\big)\Big\},$$

$$\tilde{u}_2 \in \arg\min_{u_2 \in U_2(x)} E_w\Big\{g\big(x, \tilde{u}_1, u_2, \mu_3(x) \ldots, \mu_m(x), w\big) + \alpha \tilde{J}\big(f(x, \tilde{u}_1, u_2, \mu_3(x), \ldots, \mu_m(x), w)\big)\Big\},$$

$$\cdots \qquad \cdots \qquad \cdots \qquad \cdots$$

$$\tilde{u}_m \in \arg\min_{u_m \in U_m(x)} E_w\Big\{g\big(x, \tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_{m-1}, u_m, w\big) + \alpha \tilde{J}\big(f(x, \tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_{m-1}, u_m, w)\big)\Big\}.$$

In the context of the reformulated problem, this is a sequence of one-step lookahead minimizations at the states $x, (x, \tilde{u}_0), \ldots, (x, \tilde{u}_0, \ldots, \tilde{u}_{m-1})$ of

the reformulated problem, using truncated rollout with base policy μ of corresponding length $m-1, m-2, \ldots, 0$. The Newton step interpretation of Section 3.4 and Fig. 3.4.4 still applies, with its superlinear convergence rate. At the same time, the computational requirements are dramatically reduced, similar to the multiagent rollout method discussed earlier.

Let us finally note that there are variations of the multiagent schemes of this section, which involve multistep lookahead minimization as well as truncated rollout. They likely result in better performance at the expense of greater computational cost.

3.8 ON-LINE SIMPLIFIED POLICY ITERATION

In this section, we describe some variants of the PI algorithm, introduced in the author's recent paper [Ber21a], which are consistent with the approximation in value space theme of this work. The salient feature of these variants is that *they involve an exact Newton step and are suitable for on-line implementation*, while still maintaining the principal character of PI, which we have viewed so far as an off-line algorithm.

Thus the algorithms of this section involve training and self-improvement through on-line experience. They are also simplified relative to standard PI in two ways:

(a) They perform policy improvement operations only for the states that are encountered during the on-line operation of the system.

(b) The policy improvement operation is simplified in that it uses approximate minimization in the Bellman equation of the current policy at the current state.

Despite these simplifications, we show that our algorithms generate a sequence of improved policies, which converge to a policy with a local optimality property. Moreover, with an enhancement of the policy improvement operation, which involves a form of exploration, they converge to a globally optimal policy.

The motivation comes from the rollout algorithm, which starts from some available base policy and implements on-line an improved rollout policy. In the algorithm of the present section, the data accumulated from the rollout implementation are used to improve on-line the base policy, and to asymptotically obtain a policy that is either locally or globally optimal.

We focus on a finite-state discounted Markov decision problem, with states $1, \ldots, n$, and we use a transition probability notation. We denote states by the symbol x and successor states by the symbol y. The control/action is denoted by u, and is constrained to take values in a given finite constraint set $U(x)$, which may depend on the current state x. The use of a control u at state x specifies the transition probability $p_{xy}(u)$ to the next state y, at a cost $g(x, u, y)$.

The cost of a policy μ starting from state x_0 is given by

$$J_\mu(x_0) = \lim_{N\to\infty} E\left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu(x_k), x_{k+1}\big) \;\Big|\; x_0,\, \mu \right\}, \qquad x_0 = 1, \ldots, n,$$

where $\alpha < 1$ is the discount factor. As earlier, J_μ is viewed as the vector in the n-dimensional Euclidean space $\Re^n$, with components $J_\mu(1), \ldots, J_\mu(n)$.

In terms of our abstract notation, for each policy μ, the Bellman operator T_μ maps a vector $J \in \Re^n$ to the vector $T_\mu J \in \Re^n$ that has components

$$(T_\mu J)(x) = \sum_{y=1}^{n} p_{xy}\big(\mu(x)\big)\Big(g(x, \mu(x), y) + \alpha J(y)\Big), \qquad x = 1, \ldots, n. \tag{3.23}$$

The Bellman operator $T : \Re^n \mapsto \Re^n$ is given by

$$(TJ)(x) = \min_{u\in U(x)} \sum_{y=1}^{n} p_{xy}(u)\big(g(x, u, y) + \alpha J(y)\big), \qquad x = 1, \ldots, n. \tag{3.24}$$

For the discounted problem that we consider, the operators T_μ and T are sup-norm contractions (generally this is true for discounted problems with bounded cost per stage [Ber22a]; in our context, the number of states is finite, so the cost per stage is bounded). Thus J_μ is the unique solution of Bellman's equation $J = T_\mu J$, or equivalently

$$J_\mu(x) = \sum_{y=1}^{n} p_{xy}\big(\mu(x)\big)\Big(g\big(x, \mu(x), y\big) + \alpha J_\mu(y)\Big), \qquad x = 1, \ldots, n. \tag{3.25}$$

Moreover, J^* is the unique solution of Bellman's equation $J = TJ$, so that

$$J^*(x) = \min_{u\in U(x)} \sum_{y=1}^{n} p_{xy}(u)\big(g(x, u, y) + \alpha J^*(y)\big), \qquad x = 1, \ldots, n. \tag{3.26}$$

Furthermore, the following optimality conditions hold

$$T_\mu J^* = TJ^* \qquad \text{if and only if} \qquad \mu \text{ is optimal}, \tag{3.27}$$

$$T_\mu J_\mu = TJ_\mu \qquad \text{if and only if} \qquad \mu \text{ is optimal}. \tag{3.28}$$

The contraction property also implies that the VI algorithms

$$J^{k+1} = T_\mu J^k, \qquad J^{k+1} = TJ^k$$

generate sequences $\{J^k\}$ that converge to J_μ and J^*, respectively, from any starting vector $J^0 \in \Re^n$.

As discussed earlier, in the PI algorithm, the current policy μ is improved by finding $\tilde{\mu}$ that satisfies

$$T_{\tilde{\mu}} J_\mu = T J_\mu$$

[i.e., by minimizing for all x in the right-hand side of Eq. (3.24) with J_μ in place of J]. The improved policy $\tilde{\mu}$ is evaluated by solving the linear $n \times n$ system of equations $J_{\tilde{\mu}} = T_{\tilde{\mu}} J_{\tilde{\mu}}$, and then $(J_{\tilde{\mu}}, \tilde{\mu})$ becomes the new cost vector-policy pair, which is used to start a new iteration. Thus the PI algorithm starts with a policy μ^0 and generates a sequence $\{\mu^k\}$ according to

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}, \qquad T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}. \tag{3.29}$$

We now introduce an on-line variant of PI, which starts at time 0 with a state-policy pair (x_0, μ^0) and generates on-line a sequence of state-policy pairs (x_k, μ^k). We view x_k as the current state of a system operating online under the influence of the policies $\mu^1, \mu^2, \ldots$. In our algorithm, μ^{k+1} *may differ from* μ^k *only at state* x_k. The control $\mu^{k+1}(x_k)$ and the state x_{k+1} are generated as follows:

We consider the right-hand sides of Bellman's equation for μ^k (also known as *Q-factors* of μ^k)

$$Q_{\mu^k}(x_k, u) = \sum_{y=1}^{n} p_{x_k y}(u)\big(g(x_k, u, y) + \alpha J_{\mu^k}(y)\big), \tag{3.30}$$

and we select the control $\mu^{k+1}(x_k)$ from within the constraint set $U(x_k)$ with sufficient accuracy to satisfy the following condition

$$Q_{\mu^k}\big(x_k, \mu^{k+1}(x_k)\big) \leq J_{\mu^k}(x_k), \tag{3.31}$$

with strict inequality whenever this is possible.† For $x \neq x_k$ the policy controls are not changed:

$$\mu^{k+1}(x) = \mu^k(x) \quad \text{for all } x \neq x_k.$$

The next state x_{k+1} is generated randomly according to the transition probabilities $p_{x_k x_{k+1}}\big(\mu^{k+1}(x_k)\big)$.

† By this we mean that if $\min_{u \in U(x_k)} Q_{\mu^k}(x_k, u) < J_{\mu^k}(x_k)$ we select a control u_k that satisfies

$$Q_{\mu^k}(x_k, u_k) < J_{\mu^k}(x_k), \tag{3.32}$$

and set $\mu^{k+1}(x_k) = u_k$, and otherwise we set $\mu^{k+1}(x_k) = \mu^k(x_k)$ [so Eq. (3.31) is satisfied]. Such a control selection may be obtained by a number of schemes, including brute force calculation and random search based on Bayesian optimization. The needed values of the Q-factor Q_{μ^k} and cost J_{μ^k} may be obtained in several ways, depending on the problem at hand, including by on-line simulation.

We first show that the current policy is monotonically improved, i.e., that

$$J_{\mu^{k+1}}(x) \leq J_{\mu^k}(x), \qquad \text{for all } x \text{ and } k,$$

with strict inequality for $x = x_k$ (and possibly other values of x) if

$$\min_{u \in U(x_k)} Q_{\mu^k}(x_k, u) < J_{\mu^k}(x_k).$$

To prove this, we note that the policy update is done under the condition (3.31). By using the monotonicity of $T_{\mu^{k+1}}$, we have for all $\ell \geq 1$,

$$T^{\ell+1}_{\mu^{k+1}} J_{\mu^k} \leq T^{\ell}_{\mu^{k+1}} J_{\mu^k} \leq J_{\mu^k}, \tag{3.33}$$

so by taking the limit as $\ell \to \infty$ and by using the convergence property of VI ($T^{\ell}_{\mu^{k+1}} J \to J_{\mu^{k+1}}$ for any J), we obtain $J_{\mu^{k+1}} \leq J_{\mu^k}$. Moreover, the algorithm selects $\mu^{k+1}(x_k)$ so that

$$(T_{\mu^{k+1}} J_{\mu^k})(x_k) = Q_{\mu^k}(x_k, u_k) < J_{\mu^k}(x_k)$$

if

$$\min_{u \in U(x_k)} Q_{\mu^k}(x_k, u) < J_{\mu^k}(x_k),$$

[cf. Eq. (3.32)], and then by using Eq. (3.33), we have $J_{\mu^{k+1}}(x_k) < J_{\mu^k}(x_k)$.

Local Optimality

We next discuss the convergence and optimality properties of the algorithm. We introduce a definition of local optimality of a policy, whereby the policy selects controls optimally only within a subset of states.

Given a subset of states S and a policy μ, we say that μ *is locally optimal over* S if μ is optimal for the problem where the control is restricted to take the value $\mu(x)$ at the states $x \notin S$, and is allowed to take any value $u \in U(x)$ at the states $x \in S$.

Roughly speaking, μ is locally optimal over S, if μ is acting optimally within S, but under the (incorrect) assumption that once the state of the system gets to a state x outside S, there will be no option to select control other than $\mu(x)$. Thus if the choices of μ outside of S are poor, its choices within S may also be poor.

Mathematically, μ is locally optimal over S if

$$J_\mu(x) = \min_{u \in U(x)} \sum_{y=1}^{n} p_{xy}(u)\big(g(x,u,y) + \alpha J_\mu(y)\big), \qquad \text{for all } x \in S,$$

$$J_\mu(x) = \sum_{y=1}^{n} p_{xy}\big(\mu(x)\big)\Big(g\big(x,\mu(x),y\big) + \alpha J_\mu(y)\Big), \qquad \text{for all } x \notin S,$$

which can be written compactly as

$$(T_\mu J_\mu)(x) = (TJ_\mu)(x), \qquad \text{for all } x \in S. \tag{3.34}$$

Note that this is different than (global) optimality of μ, which holds if and only if the above condition holds for all $x = 1, \ldots, n$, rather than just for $x \in S$ [cf. Eq. (3.28)]. However, it can be seen that a (globally) optimal policy is also locally optimal within any subset of states.

Our main convergence result is the following.

Proposition 3.8.1: Let $\overline{S}$ be the subset of states that are repeated infinitely often within the sequence $\{x_k\}$. Then the corresponding sequence $\{\mu^k\}$ converges finitely to some policy $\overline{\mu}$ in the sense that $\mu^k = \overline{\mu}$ for all k after some index $\overline{k}$. Moreover $\overline{\mu}$ is locally optimal within $\overline{S}$, while $\overline{S}$ is invariant under $\overline{\mu}$, in the sense that

$$p_{xy}\big(\overline{\mu}(x)\big) = 0 \qquad \text{for all } x \in \overline{S} \text{ and } y \notin \overline{S}.$$

Proof: The cost function sequence $\{J_{\mu^k}\}$ is monotonically nonincreasing, as shown earlier. Moreover, the number of policies μ is finite in view of the finiteness of the state and control spaces. Therefore, the number of corresponding functions J_μ is also finite, so J_{μ^k} converges in a finite number of steps to some $\bar{J}$, which in view of the algorithm's construction [selecting $u_k = \mu^k(x_k)$ if $\min_{u \in U(x_k)} Q_{\mu^k}(x_k, u) = J_{\mu^k}(x_k)$; cf. Eq. (3.32)], implies that μ^k will remain unchanged at some $\overline{\mu}$ with $J_{\overline{\mu}} = \bar{J}$ after some sufficiently large k.

We will show that the local optimality condition (3.34) holds for $S = \overline{S}$ and $\mu = \overline{\mu}$. In particular, we have $x_k \in \overline{S}$ and $\mu^k = \overline{\mu}$ for all k greater than some index, while for every $x \in \overline{S}$, we have $x_k = x$ for infinitely many k. It follows that for all $x \in \overline{S}$,

$$Q_{\overline{\mu}}\big(x, \overline{\mu}(x)\big) = J_{\overline{\mu}}(x), \tag{3.35}$$

while by the construction of the algorithm,

$$Q_{\overline{\mu}}\big(x, u\big) \geq J_{\overline{\mu}}(x), \qquad \text{for all } u \in U(x), \tag{3.36}$$

since the reverse would imply that $\mu^{k+1}(x) \neq \mu^k(x)$ for infinitely many k [cf. Eq. (3.32)]. Condition (3.35) can be written as $J_{\overline{\mu}}(x) = (T_{\overline{\mu}} J_{\overline{\mu}})(x)$ for all $x \in \overline{S}$, and combined with Eq. (3.36), implies that

$$(T_{\overline{\mu}} J_{\overline{\mu}})(x) = (TJ_{\overline{\mu}})(x), \qquad \text{for all } x \in \overline{S}.$$

This is the local optimality condition (3.34) with $S = \overline{S}$ and $\mu = \overline{\mu}$.

To show that $\overline{S}$ is invariant under $\overline{\mu}$, we argue by contradiction: if this were not so, there would exist a state $x \in \overline{S}$ and a state $y \notin \overline{S}$ such that $p_{xy}\big(\overline{\mu}(x)\big) > 0$, implying that y would be generated following the occurrence of x infinitely often within the sequence $\{x_k\}$, and hence would have to belong to $\overline{S}$ (by the definition of $\overline{S}$). **Q.E.D.**

Note an implication of the invariance property of the set $\overline{S}$ shown in the preceding proposition. We have that $\overline{\mu}$ is (globally) optimal under the assumption that for every policy there does not exist any strict subset of states that is invariant.

A Counterexample to Global Optimality

The following deterministic example (given to us by Yuchao Li) shows that the policy $\overline{\mu}$ obtained by the algorithm need not be (globally) optimal. Here there are three states 1, 2, and 3. From state 1 we can go to state 2 at cost 1, and to state 3 at cost 0, from state 2 we can go to states 1 and 3 at cost 0, and from state 3 we can go to state 2 at cost 0 or stay in 3 at a high cost (say 10). The discount factor is $\alpha = 0.9$. Then it can be verified that the optimal policy is

$$\mu^*(1) : \text{Go to 3}, \qquad \mu^*(2) : \text{Go to 3}, \qquad \mu^*(3) : \text{Go to 2},$$

with optimal costs

$$J^*(1) = J^*(2) = J^*(3) = 0,$$

while the policy

$$\overline{\mu}(1) : \text{Go to 2}, \qquad \overline{\mu}(2) : \text{Go to 1}, \qquad \overline{\mu}(3) : \text{Stay at 3},$$

is strictly suboptimal, but is locally optimal over the set of states $\overline{S} = \{1, 2\}$. Moreover our on-line PI algorithm, starting from state 1 and the policy $\mu^0 = \overline{\mu}$, oscillates between the states 1 and 2, and leaves the policy μ^0 unchanged. Note also that $\overline{S}$ is invariant under $\overline{\mu}$, consistent with Prop. 3.8.1.

On-Line Variants of Policy Iteration with Global Optimality Properties

To address the local versus global convergence issue illustrated by the preceding example, we consider an alternative scheme, whereby in addition to u_k, we generate an additional control at a randomly chosen state $\overline{x}_k \neq x_k$.† In particular, assume that at each time k, in addition to u_k and x_{k+1} that

† It is also possible to choose multiple additional states at time k for a policy improvement operation, and this is well-suited for the use of parallel computation.

are generated according to Eq. (3.32), the algorithm generates randomly another state $\overline{x}_k$ (all states are selected with positive probability), performs a policy improvement operation at that state as well, and modifies accordingly $\mu^{k+1}(\overline{x}_k)$. Thus, in addition to a policy improvement operation at each state within the generated sequence $\{x_k\}$, there is an additional policy improvement operation at each state within the randomly generated sequence $\{\overline{x}_k\}$.

Because of the random mechanism of selecting $\overline{x}_k$, it follows that at every state there will be a policy improvement operation infinitely often, which implies that the policy $\overline{\mu}$ ultimately obtained is (globally) optimal. Note also that *we may view the random generation of the sequence* $\{\overline{x}_k\}$ *as a form of exploration*. The probabilistic mechanism for generating the random sequence $\{\overline{x}_k\}$ may be guided by some heuristic reasoning, which aims to explore states with high cost improvement potential.

Let us also note the possibility of approximate implementations of the algorithms described above. In particular, one may start with some base policy, which may be periodically updated using some approximation in policy space scheme, while incorporating the policy improvement data generated so far. As long as the most recent policy improvement results are maintained for the states that have been encountered in the past, the convergence results described above will be maintained.

Finally, let us mention that the idea of on-line PI of the present section can be extended to a broader algorithmic context of *on-line improvement of the approximation in value space process*. In particular, we may consider starting the on-line play algorithm with a cost function approximation $\tilde{J}$, obtained through some off-line training process. We may then try to gradually enhance the quality of $\tilde{J}$ through on-line experience. For example, $\tilde{J}$ may be constructed through some form of machine learning or Bayesian optimization method that is capable of improving the approximation using data obtained in the process of on-line play. There are many possibilities along these lines, and they are a fruitful area of research, particularly within the context of specific applications.

3.9 EXCEPTIONAL CASES

Let us now consider situations where exceptional behavior occurs. One such situation is when the Bellman equation $J = TJ$ has multiple solutions. Then the VI algorithm, when started at one of these solutions will stay at that solution. More generally, it may be unclear whether the VI algorithm will converge to J^*, even when started at seemingly favorable initial conditions. Other types of exceptional behavior may also occur, including cases where the Bellman equation has no solution within the set of real-valued functions. The most unusual case of all is when J^* is real-valued but does not satisfy the Bellman equation $J = TJ$, which in turn has other real-valued solutions; see [BeY16] and [Ber22a], Section 3.1. This is a

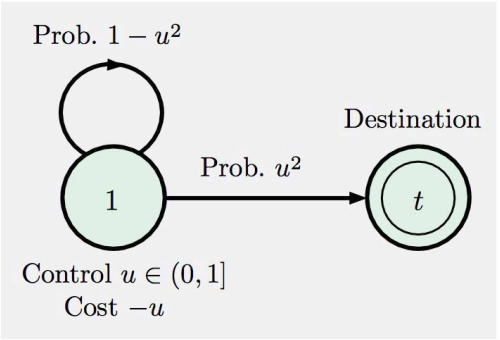

Figure 3.9.1. Transition diagram for the blackmailer problem. At state 1, the blackmailer may demand any amount $u \in (0, 1]$. The victim will comply with probability $1 - u^2$ and will not comply with probability u^2, in which case the process will terminate.

highly unusual phenomenon, which will not be discussed here. It need not be of practical concern, as it arises only in artificial examples; see [BeY16]. Still it illustrates the surprising range of exceptional situations that should be taken into account in theoretical analyses and computational studies.

In this section we provide some examples that illustrate the mechanism by which exceptional behavior in infinite horizon DP can occur, and we highlight the need for rigorous analysis of RL methods when used in contexts that are beyond the well-behaved discounted case, where the Bellman operator is a contraction mapping. For further discussion and analysis that address exceptional behavior, including the frameworks of semicontractive and noncontractive DP, we refer to the author's abstract DP monograph [Ber22a].

The Blackmailer's Dilemma

This is a classical example involving a profit maximizing blackmailer. We formulate it as an SSP problem involving cost minimization, with a single state $x = 1$, in addition to the termination state t. We are in state 1 when the victim is compliant, and we are in state t when the victim refuses to yield to the blackmailer's demand (a refusal is permanent, in the sense that once the blackmailer's demand is refused, all subsequent demands are assumed to be refused, so t is a termination state). At state 1 we can choose a control $u \in (0, 1]$, which we regard as the demand made by the blackmailer. The problem is to find the blackmailer's policy that maximizes his expected total gain.

To formulate this problem as a minimization problem, we will use $(-u)$ as the cost per stage. In particular, upon choosing $u \in (0, 1]$, we move to state t with probability u^2, and stay in state 1 with probability $1 - u^2$; see Fig. 3.9.1. The idea is to optimally balance the blackmailer's

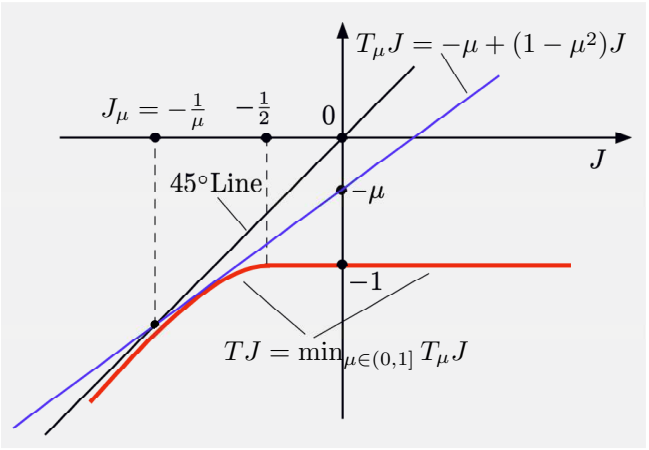

Figure 3.9.2. The Bellman operators and the Bellman equation for the blackmailer problem.

desire for increased demands (large u) with keeping his victim compliant (small u).

For notational simplicity, let us abbreviate $J(1)$ and $\mu(1)$ with just the scalars J and μ, respectively. Then in terms of abstract DP we have $X = \{1\}$, $U = (0, 1]$, and for every stationary policy μ, the corresponding Bellman operator T_μ, restricted to state 1 is given by

$$T_\mu J = -\mu + (1 - \mu^2)J; \tag{3.37}$$

[at the state t, $J_\mu(t) = 0$]. Clearly T_μ is linear, maps the real line $\Re$ to itself, and is a contraction with modulus $1 - \mu^2$. Its unique fixed point within $\Re$, J_μ, is the solution of

$$J_\mu = T_\mu J_\mu = -\mu + (1 - \mu^2)J_\mu,$$

which yields

$$J_\mu = -\frac{1}{\mu};$$

see Fig. 3.9.2. Here all policies are stable and lead asymptotically to t with probability 1, and the infimum of J_μ over $\mu \in (0, 1]$ is $-\infty$, implying also that $J^* = -\infty$. However, there is no optimal policy.

The Bellman operator T is given by

$$TJ = \min_{0<u\leq 1} \big\{ -u + (1 - u^2)J \big\},$$

which after some calculation can be shown to have the form

$$TJ = \begin{cases} -1 & \text{for } -\frac{1}{2} \leq J, \\ J + \frac{1}{4J} & \text{for } J \leq -\frac{1}{2}. \end{cases}$$

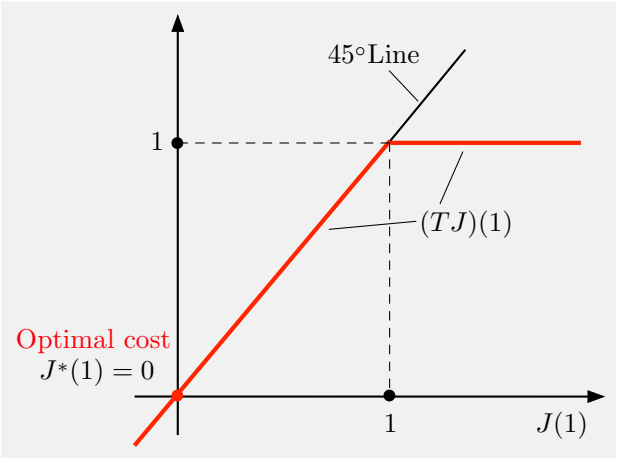

Figure 3.9.3 Illustration of the Bellman equation for a shortest path problem in the exceptional case where there is a cycle of zero length. Restricted within the set of J with $J(t) = 0$, the Bellman operator has the form

$$(TJ)(1) = \min\big\{J(1),\, 1\big\}.$$

The set of solutions of Bellman's equation, $J(1) = (TJ)(1)$ is the interval $(-\infty, 1]$ and contains $J^*(1) = 0$ in its interior.

The form of T is illustrated in Fig. 3.9.2. It can be seen from this figure that the Bellman equation $J = TJ$ has no real-valued solution (the optimal cost $J^* = -\infty$ is a solution within the set of extended real numbers $[-\infty, \infty]$). Moreover the VI algorithm will converge to J^* starting from any $J \in \Re$. It can be verified also that the PI algorithm, starting from any policy $\mu^0 \in (0, 1]$, produces the ever improving sequence of policies $\{\mu^k\}$ with $\mu^{k+1} = \mu^k/2$. Thus μ^k converges to 0, which is not a feasible policy. Also $J_{\mu^k} = -1/\mu^k$, and we have $J_{\mu^k} \downarrow -\infty = J^*$, so the PI algorithm gives in the limit the infinite optimal cost. For additional related examples and discussion relating to the blackmailer problem, see [Ber22a], Section 3.1.

A Shortest Path Problem

Another exceptional type of example is provided by shortest path problems that contain cycles of zero length; see the monograph [Ber22a], Section 3.1. In this case there are infinitely many solutions to Bellman's equation, and the VI and PI algorithms, as well as the approximation in value space process exhibit unusual behavior. We demonstrate this with a shortest path problem involving a single state, denoted 1, in addition to the cost-free destination state t.

In particular, let $X = \{t, 1\}$, and assume that at state 1 there are two options: we can stay at 1 at cost 0, or move to t at cost 1. Here

$J^*(t) = J^*(1) = 0$, and there are just two policies, which correspond to the two options at state 1 and are stable. The optimal policy starting at state 1 is to stay at 1. If we restrict attention to cost functions J with $J(t) = 0$, the Bellman operator is

$$(TJ)(1) = \min\big\{J(1),\, 1\big\},$$

and Bellman's equation, written as an equation in $J(1)$, has the form

$$J(1) = \min\big\{J(1),\, 1\big\}.$$

The set of solutions of this equation is the interval $(-\infty, 1]$ and it is infinite; see Fig. 3.9.3. The optimal value $J^*(1) = 0$ lies in the interior of this set, and cannot be obtained by the VI algorithm, unless the algorithm is started at the optimal value.

Let us consider approximation in value space with cost approximation $\tilde{J}(1)$. Then it can be seen that if $\tilde{J}(1) < 1$, the one-step lookahead policy is to stay at state 1, which is optimal. If $\tilde{J}(1) > 1$, the one-step lookahead policy is to move from state 1 to state t, which is suboptimal. If $\tilde{J}(1) = 1$, either one of the two policies can be the one-step lookahead policy.

Consider also the PI algorithm, starting from the suboptimal policy μ that moves from state 1 to state t. Then $J_\mu(t) = 0$, $J_\mu(1) = 1$, and it can be seen that μ satisfies the policy improvement equation

$$\mu(1) \in \arg\min\big\{J_\mu(1),\, 1 + J_\mu(t)\big\}$$

(the same is true for the optimal policy that stays at state 1). Thus the PI algorithm may stop with the suboptimal policy μ.

Problems where exceptional behavior occurs arise often in Markov decision problems, once one departs from the most commonly discussed and best behaved paradigm of discounted cost with bounded cost per stage, where the mappings T_μ of all policies μ have favorable contraction properties. Moreover, problems arising in decision and control, such as those that have been addressed with MPC, often give rise to exceptional behavior. Further research and computational experimentation is expected to provide improved guidelines for the solution of such problems.

What Happens When the Bellman Operator is Neither Concave nor Convex? - Markov Games

We have discussed so far DP models where the Bellman operator has a concavity property. On the other hand there are interesting DP models where this is not so. An important case in point is *discounted Markov games*, a form of zero-sum games with a dynamic Markov chain structure.

Let us consider two players that play repeated matrix games at each of an infinite number of stages, using mixed strategies. The game played

at a given stage is defined by a state x that takes values in a finite set X, and changes from one stage to the next according to a Markov chain whose transition probabilities are influenced by the players' choices. At each stage and state $x \in X$, the minimizer selects a probability distribution $u = (u_1, \ldots, u_n)$ over n possible choices $i = 1, \ldots, n$, and the maximizer selects a probability distribution $v = (v_1, \ldots, v_m)$ over m possible choices $j = 1, \ldots, m$. If the minimizer chooses i and the maximizer chooses j, the payoff of the stage is $a_{ij}(x)$ and depends on the state x. Thus the expected payoff of the stage is $\sum_{i,j} a_{ij}(x) u_i v_j$ or $u'A(x)v$, where $A(x)$ is the $n \times m$ matrix with components $a_{ij}(x)$ (u and v are viewed as column vectors, and a prime denotes transposition). The two players choose u and v with knowledge of the state x, so they are viewed as using policies μ and ν, where $\mu(x)$ and $\nu(x)$ are the choices of the minimizer and the maximizer, respectively, at a state x.

The state evolves according to transition probabilities $q_{xy}(i, j)$, where i and j are the moves selected by the minimizer and the maximizer, respectively (here y represents the next state and game to be played after moves i and j are chosen at the game represented by x). When the state is x, under u and v, the state transition probabilities are

$$p_{xy}(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{m} u_i v_j q_{xy}(i, j) = u' Q_{xy} v,$$

where Q_{xy} is the $n \times m$ matrix that has components $q_{xy}(i, j)$. Payoffs are discounted by $\alpha \in (0, 1)$, and the objectives of the minimizer and maximizer, are to minimize and to maximize the total discounted expected payoff, respectively.

It was shown by Shapley [Sha53] that the problem can be formulated as a fixed point problem involving the mapping H given by

$$\begin{aligned} H(x, u, v, J) &= u' A(x) v + \alpha \sum_{y \in X} p_{xy}(u, v) J(y) \\ &= u' \left(A(x) + \alpha \sum_{y \in X} Q_{xy} J(y) \right) v, \end{aligned} \tag{3.38}$$

with the corresponding Bellman operator given by

$$(TJ)(x) = \min_{u \in U} \max_{v \in V} H(x, u, v, J), \qquad \text{for all } x \in X. \tag{3.39}$$

It can be verified that T is an unweighted sup-norm contraction, and its unique fixed point J^* satisfies the Bellman equation $J^* = TJ^*$.

Note that since the matrix defining the mapping H of Eq. (3.38),

$$A(x) + \alpha \sum_{y \in X} Q_{xy} J(y),$$

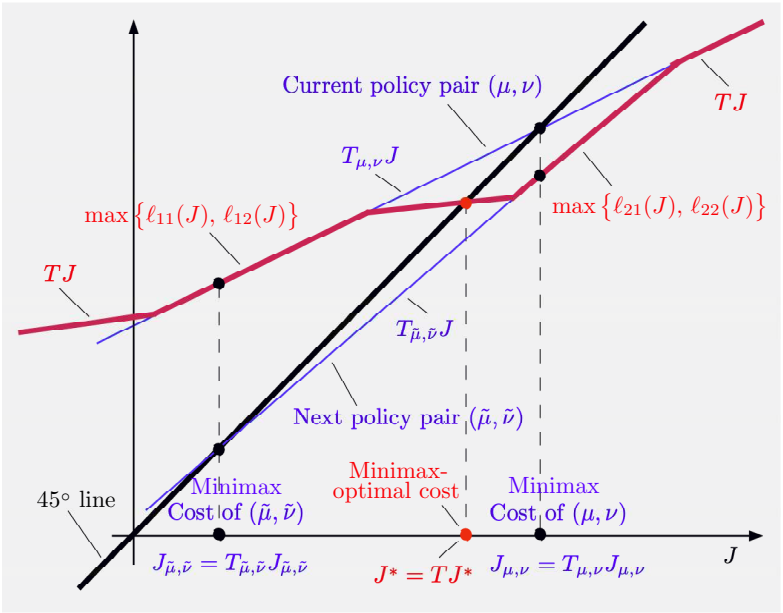

Figure 3.9.4 Schematic illustration of the PI algorithm/Newton's method in the case of a Markov game involving a single state, in addition to a termination state t. We have $J^*(t) = 0$ and $(TJ)(t) = 0$ for all J with $J(t) = 0$, so that the operator T can be graphically represented in just one dimension (denoted by J) that corresponds to the nontermination state. This makes it easy to visualize T and geometrically interpret why Newton's method does not converge. Because the operator T may be neither convex nor concave for a Markov game, the algorithm may cycle between pairs (μ, ν) and $(\tilde{\mu}, \tilde{\nu})$, as shown in the figure. By contrast in a (single-player) finite-state Markov decision problem, $(TJ)(x)$ is piecewise linear and concave, and the PI algorithm converges in a finite number of iterations.

The figure illustrates an operator T of the form

$$TJ = \min\Big\{ \max\big\{\ell_{11}(J),\, \ell_{12}(J)\big\},\, \max\big\{\ell_{21}(J),\, \ell_{22}(J)\big\}\Big\},$$

where $\ell_{ij}(J)$, are linear functions of J, corresponding to the choices $i = 1, 2$ of the minimizer and $j = 1, 2$ of the maximizer. Thus TJ is the minimum of the convex functions

$$\max\big\{\ell_{11}(J),\, \ell_{12}(J)\big\} \qquad \text{and} \qquad \max\big\{\ell_{21}(J),\, \ell_{22}(J)\big\},$$

as shown in the figure. Newton's method linearizes TJ at the current iterate [i.e., replaces TJ with one of the four linear functions $\ell_{ij}(J)$, $i = 1, 2$, $j = 1, 2$ (the one attaining the min-max at the current iterate)] and solves the corresponding linear fixed point problem to obtain the next iterate. The figure illustrates a case where the PI algorithm/Newton's method oscillates between two pairs of policies (μ, ν) and $(\tilde{\mu}, \tilde{\nu})$.

is independent of u and v, we may view $J^*(x)$ as the value of a static

(nonsequential) matrix game that depends on x. In particular, from a fundamental saddle point theorem for matrix games, we have

$$\min_{u\in U}\max_{v\in V} H(x,u,v,J^*) = \max_{v\in V}\min_{u\in U} H(x,u,v,J^*), \qquad \text{for all } x\in X. \tag{3.40}$$

The paper by Shapley [Sha53] also showed that the strategies obtained by solving the static saddle point problem (3.40) correspond to a saddle point of the sequential game in the space of mixed strategies. Thus once we find J^* as the fixed point of the mapping T [cf. Eq. (3.39)], we can obtain equilibrium policies for the minimizer and maximizer by solving the matrix game (3.40). Moreover, T can be defined via the operator $T_{\mu,\nu}$ defined for a pair of minimizer-maximizer policies (μ,ν) by

$$(T_{\mu,\nu}J)(x) = H\big(x,\mu(x),\nu(x),J\big), \qquad \text{for all } x\in X, \tag{3.41}$$

In particular, T can be defined via a minimax operation applied to the operator $T_{\mu,\nu}$ as follows:

$$(TJ)(x) = \min_{\mu\in\mathcal{M}}\max_{\nu\in\mathcal{N}}(T_{\mu,\nu}J)(x), \qquad \text{for all } x\in X,$$

where $\mathcal{M}$ and $\mathcal{N}$ are the sets of policies of the minimizer and the maximizer, respectively.

On the other hand the Bellman operator components $(TJ)(x)$ may be neither convex nor concave. In particular, the maximization makes the function

$$\max_{v\in V} H(x,u,v,J)$$

convex as a function of x for each fixed $u\in U$, while the subsequent minimization over $u\in U$ tends to introduce concave "pieces" into $(TJ)(x)$. It is possible to apply PI ideas and the corresponding Newton's method to compute the fixed point of T, and in fact this has been proposed by Pollatschek and Avi-Itzhak [PoA69]. However, this algorithm need not converge to the optimal and may not yield J^*, the fixed point of T (unless the starting point is sufficiently close to J^*, as has been recognized in the paper [PoA69]). The mechanism by which this phenomenon may occur is illustrated in Fig. 3.9.4. In fact a two-state example where the PI algorithm/Newton's method does not converge to J^* was given by van der Wal [Van78]. The preceding Markov chain discussion is part of a broader investigation of abstract minimax problems and Markov games, given in the author's recent paper [Ber21c] (and replicated in the monograph [Ber22a], Ch. 5). In particular, this paper develops exact and approximate PI methods, which correct the exceptional behavior illustrated in Fig. 3.9.4.

3.10 NOTES AND SOURCES

The author's abstract DP monograph [Ber22a] (originally published in 2013, with a second edition appearing in 2018, and a third edition appearing

in 2022) has provided the framework for the Newton step interpretations and visualizations that we have used to gain insight into approximation in value space, rollout, and policy iteration. The abstract framework aims at a unified development of the core theory and algorithms of total cost sequential decision problems, and addresses simultaneously stochastic, minimax, game, risk-sensitive, and other DP problems, through the use of the abstract DP operator (or Bellman operator as it is often called in RL). The idea here is to gain insight through abstraction. In particular, the structure of a DP model is encoded in its abstract Bellman operator, which serves as the "mathematical signature" of the model. Characteristics of this operator (such as monotonicity and contraction) largely determine the analytical results and computational algorithms that can be applied to that model.

Abstraction also captures the generality of the DP methodology. In particular, our conceptual framework based on Newton's method is applicable to problems with general state and control spaces, ranging from the continuous spaces control problems, traditionally the focus of MPC, to Markov decision problems, traditionally the focus of operations research as well as RL, and to discrete optimization problems, traditionally the focus of integer programming and combinatorial optimization. A key mathematical fact in this respect is that while the state and control spaces may be continuous or discrete, the Bellman operators and equations are always defined over continuous function spaces, and are thus amenable to solution through the use of continuous spaces algorithms, including Newton's method.