Chapter 5

Infinite Horizon Models: Expected Discounted Reward

I early inquired the rate of interest on invested money, and worried my child's brain into an understanding of the virtues and excellencies of that remarkable invention of man, compound interest[1].

Jack London, American author, 1876-1916.

Chapter 5 concerns *infinite horizon Markov decision processes under the expected discounted reward criterion*. To avoid the above long-winded expression, they are referred to simply as *discounted models*.

The discounted model provides a gold standard for theory and algorithms in infinite horizon models. Unlike undiscounted models, most results do not depend on the underlying Markov chain structure. This is because the discount factor $0 \leq \lambda < 1$ dampens out the limiting behavior of the system since $\lambda^n \to 0$ and $n \to \infty$. Consequently, theory and methods draw from linear algebra and analysis, requiring little probability theory.

As noted in Section 2.3.2, discounting is usually interpreted as incorporating the time value of money when making decisions: a reward of one dollar next year is equivalent to $0 \leq \lambda < 1$ dollars today. You may already be familiar with the concept of an *interest rate*, which accounts for the time value of money in the forward direction. If you invest one dollar today and the (risk-less) annual interest rate equals i, then you

[1] London [1906].

would have $1+i$ dollars next year, $(1+i)^2$ dollars in two years, etc. This phenomenon is referred to as compounding. In relation to interest, the discount rate is $\lambda = 1/(1+i)$. That is, receiving one dollar a year from now is equivalent to investing λ dollars now. In other words the discount factor accounts for the time value of money in the backwards direction. See Figure 5.1. Note also that discounted models are equivalent to infinite horizon models with the expected total reward criterion that terminate at an independent geometric stopping time with parameter $1-\lambda$.

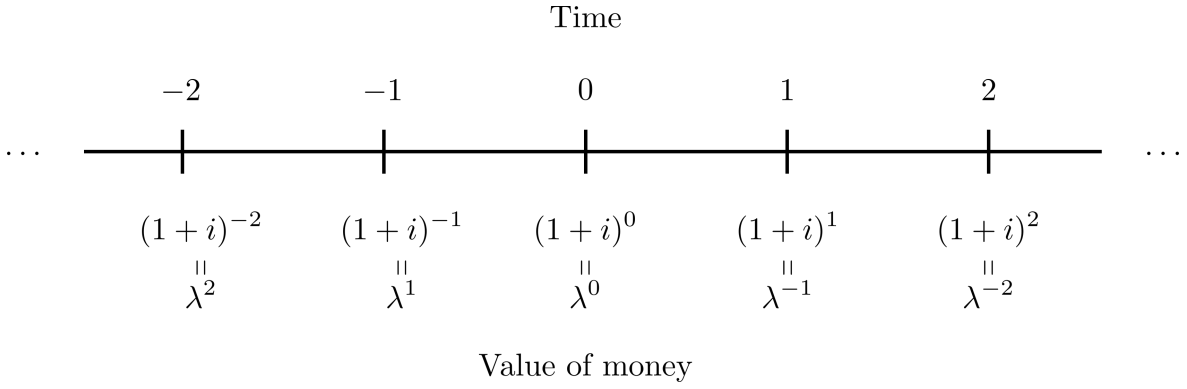


Figure 5.1: This diagram depicts the relationship between compound interest and discounting. From the compound interest perspective, \$1 invested at time 0 at interest rate i will increase to $(1+i)^2$ two-periods in the future ($t=2$). Conversely, receiving \$1 now is equivalent to investing λ^2 dollars at time $t=-2$ or in other words, an investor would be indifferent between receiving 1 two-periods in the future and possessing λ^2 now.

This chapter unfolds as follows. In several steps, culminating with establishing the existence and uniqueness of the solution to the Bellman equation, it establishes the optimality of stationary deterministic policies in the class of history-dependent randomized policies. The chapter then analyzes three classes of iterative algorithms:

- Value iteration
- Policy iteration
- Modified policy iteration

In each case, it describes the algorithm and its variants, provides stopping rules, proves convergence and presents illustrative examples. It then explores the equivalence between discounted Markov decision processes and linear programming. Finally it revisits several applications and shows how to extend the results on structured optimal policies in Section 4.4 to discounted models.

This and subsequent chapters use vector and matrix notation and rely on vector space concepts. We encourage the reader to carefully go through this chapter before

studying other optimality criteria, approximate dynamic programming, simulation-based methods, and reinforcement learning.

5.1 Preliminaries

This section introduces notation and basic recursions.

5.1.1 Reward functions

Recall that the reward function $r_n(s, a, j)$ equals the reward received between decision epoch n and $n + 1$ when action a is chosen in state s at decision epoch n, and a transition occurs to state j prior to decision epoch $n + 1$. In infinite horizon models under an expected reward based criterion, equations may appear more familiar and transparent if they are expressed in terms of $r_n(s, a)$ instead of $r_n(s, a, j)$. This may also be the case when the reward does not depend on j such as in the queuing control models in Section 3.4. When $r_n(s, a, j)$ is a model primitive, the expected reward from choosing action a in state s at decision epoch n can be written

$$r_n(s, a) = \sum_{j \in S} r_n(s, a, j) p_n(j|s, a). \tag{5.1}$$

This calculation in the two-state model in Section 2.5 as follows.

Example 5.1. Assume the rewards and probabilities are stationary. Then,

$$\begin{aligned} r(s_1, a_{1,1}) &= r(s_1, a_{1,1}, s_1)p(s_1|s_1, a_{1,1}) + r(s_1, a_{1,1}, s_2)p(s_2|s_1, a_{1,1}) \\ &= 5 \cdot 0.8 - 5 \cdot 0.2 = 3 \end{aligned}$$

The remaining rewards can be calculated similarly and shown to be given by $r(s_1, a_{1,2}) = 5$, $r(s_2, a_{2,1}) = -5$ and $r(s_2, a_{2,2}) = 2$.

This change in notation is primarily for convenience; it does not affect the principles of our development of infinite horizon Markov decision processes. One consequence of this change in the form of the reward function is that the recursions and Bellman equation will look slightly different from the ones presented in the previous chapter. The chapters on simulation methods (Chapters 10 and 11) retain $r_n(s, a, j)$ as a primitive because simulation replaces expectations by simulated sample paths.

5.1.2 Key assumptions

In addition to the standard assumptions of i finite states and finite actions, the following additional assumptions hold throughout this chapter:

Stationary rewards: The reward function does not vary from epoch to epoch. It will be written as $r(s, a)$, independent of n.

Bounded rewards: There is a finite W such that $|r(s, a)| \leq W$ for all $a \in A_s, s \in S$.

Stationary transition probabilities: The transition probabilities will be written $p(j|s, a)$, independent of n.

Discounting: Future rewards are discounted by a multiplicative factor λ per period, where $0 \leq \lambda < 1$.

5.1.3 Markovian policies are sufficient

Chapter 4 (Theorem 4.2) showed that Markovian deterministic policies are optimal among the class of history-dependent randomized policies for finite horizon Markov decision process. A stronger result holds in the infinite horizon setting: stationary deterministic policies, a subset of Markovian deterministic policies, are optimal within the class of history-dependent randomized policies.

To simplify exposition, this chapter considers Markovian randomized policies instead of history-dependent randomized policies. The following result formally justifies this simplification. A proof of it appears in Appendix A at the end of this chapter. It is the consequence of technical Lemma 5.9, which establishes that given any history-dependent policy, for each initial state, there exists a Markovian randomized policy that has the same action selection probabilities at each decision epoch.

Theorem 5.1. For each $s \in S$, given any policy $\pi = (d_1, d_2, \ldots) \in \Pi^{\mathrm{HR}}$, there exists a policy $\pi' = (d'_1, d'_2, \ldots) \in \Pi^{\mathrm{MR}}$ with the same expected discounted reward conditional on $X_1 = s$.

5.1.4 Matrix and vector representation for stationary policies

To streamline results and simplify many proofs and formulae, value functions and rewards will often be represented by (column) vectors and transition probabilities by matrices. Vectors and matrices are denoted by **bold** symbols. Key notation is defined below, with all definitions assuming S finite. All concepts can be extended to S countable, continuous or abstract, but such generality will not be needed here.

Matrix and vector notation

V: Vector or linear space[a] of real-valued $|S|$-dimensional vectors.

$\mathbf{v}$: $|S|$-dimensional real-valued vector usually representing a value function or an approximation.

$\mathbf{r}_d$: $|S|$-dimensional vector of rewards corresponding to $d \in D^{\text{MR}}$ (see equation (5.2)).

$\mathbf{e}$: Vector with all components equal to one.

$\mathbf{e}_s$: Vector with a 1 in the s-th component and zeroes elsewhere.

$\mathbf{0}$: Vector with all components equal to zero.

$\mathbf{P}_d$: $|S| \times |S|$-dimensional transition probability matrix corresponding to $d \in D^{\text{MR}}$ (see equation (5.3)).

$\mathbf{I}$: $|S| \times |S|$-dimensional identity matrix.

[a]For our purposes a vector or linear space is a set of elements closed under addition and scalar multiplication. In addition, several other technical properties hold.

Note that $\mathbf{v}$, $\mathbf{r}_d$, $\mathbf{e}$, $\mathbf{e}_s$ and $\mathbf{0}$ are elements of V and the matrices $\mathbf{P}_d$ and $\mathbf{I}$ map $V \to V$.

Some further notational conventions follow. Be sure you understand them before proceeding through the technical parts of this chapter. Assume for concreteness that $S = \{s_1, \ldots, s_M\}$ is M-dimensional. The vector $\mathbf{v}$ has components $v(s_1), \ldots, v(s_M)$. When written inline, $\big(v(s_1), \ldots, v(s_M)\big)$ will denote the column vector $\mathbf{v}$. The transpose of any vector $\mathbf{v}$ will be written $\mathbf{v}^\mathsf{T}$. So $(v(s_1), \ldots, v(s_M))^\mathsf{T}$ represents a row vector. Note that components of a vector are *not* expressed in bold. However, when multiplying a vector $\mathbf{v}$ by a matrix $\mathbf{P}_d$, $\mathbf{P}_d\mathbf{v}(s_k)$ denoted the s_k-th component of the resulting vector. Notice that the vector $\mathbf{v}$ remains bold here, since the quantity of interest is not the s_k-th component of $\mathbf{v}$, but of $\mathbf{P}_d\mathbf{v}$.

Operators

We find it convenient, transparent and elegant to use *operators* to express fundamental Markov decision process relationships. From the perspective of this book, an operator $T : V \to V$ is a (possibly nonlinear) function that assigns the value $T\mathbf{v} \in V$ to each vector $\mathbf{v} \in V$. When V is M-dimensional, $T\mathbf{v}$ has M components. Moreover, in contrast to how functions are written as $f(x)$, parentheses are not used when writing

$T\mathbf{v}$[2]. Define $T\mathbf{v}(s)$ as the s-th component of $T\mathbf{v}$ and note that $T^n\mathbf{v} = T\cdots T\mathbf{v}$.

Rewards and transition probabilities corresponding to Markovian policies

Next, consider $\mathbf{r}_d$ and $\mathbf{P}_d$ for deterministic and randomized Markovian decision rules. The s-th component of $\mathbf{r}_d$, denoted by $r_d(s)$, satisfies

$$r_d(s) = \begin{cases} r(s, d(s)) & d \in D^{\mathrm{MD}} \\ \sum_{a\in A_s} w_d(a|s) r(s,a) & d \in D^{\mathrm{MR}}. \end{cases} \tag{5.2}$$

Similarly, the (s,j)-th component of $\mathbf{P}_d$ denoted by $P_d(j|s)$, equals

$$P_d(j|s) = \begin{cases} p(j|s, d(s)) & d \in D^{\mathrm{MD}} \\ \sum_{a\in A_s} w_d(a|s) p(j|s,a) & d \in D^{\mathrm{MR}}. \end{cases} \tag{5.3}$$

Note that if the model had been formulated in terms of $r(s,a,j)$, $\mathbf{r}_d$ would be replaced by a matrix[3].

Given a policy $\pi = (d_1, d_2, \ldots) \in \Pi^{\mathrm{MR}}$, the n-step transition probability matrix $\mathbf{P}_\pi^n$ is

$$\mathbf{P}_\pi^n := \mathbf{P}_{d_1}\mathbf{P}_{d_2}\cdots\mathbf{P}_{d_n} \tag{5.4}$$

with its (s,j)-th component representing $p^\pi(X_n = j | X_1 = s)$.

Since V is a vector space, it is closed under addition and scalar multiplication, so that for $\mathbf{v} \in V$, $\mathbf{P}_d\mathbf{v} \in V$, $\lambda\mathbf{P}_d\mathbf{v} \in V$, and most importantly, $\mathbf{r}_d + \lambda\mathbf{P}_d\mathbf{v} \in V$.

5.1.5 Partial orders

To express the Bellman equation in vector notation, the "max" over a set of vectors must be defined carefully. Since the definition of a vector space does not include any notion of an ordering of vectors, the concept of a *partial order* is needed, in particular the *component-wise* partial order. This means that $\mathbf{v} \geq \mathbf{u}$ if $v(s) \geq u(s)$ for all $s \in S$.

Given two vectors, $\mathbf{v}$ and $\mathbf{u}$, there are four possibilities: $\mathbf{v} \geq \mathbf{u}$, $\mathbf{u} \geq \mathbf{v}$, $\mathbf{u} = \mathbf{v}$ or they are not comparable. An example of vectors that are not comparable are $\mathbf{v} = (1,2)$ and $\mathbf{u} = (2,1)$, since they are not equal, nor is one larger (component-wise) than the other. Because there exist incomparable vectors, this is a partial order, rather than a total order.

For the component-wise partial order, a corresponding component-wise maximum is needed.

[2]This approach to expressing operators generalizes the approach used to indicate a product of a matrix (a linear operator) and a vector. If $\mathbf{A}$ is a matrix and $\mathbf{v}$ a vector, it is standard to write $\mathbf{Av}$ instead of $\mathbf{A}(\mathbf{v})$.

[3]If $d \in D^{\mathrm{MD}}$, $\mathbf{R}_d$ would represent the matrix with components $r(s, d(s), j)$ with an obvious generalization to $d \in D^{\mathrm{MR}}$. This representation will not be used in the sequel.

Definition 5.1. The *component-wise maximum* of a set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_K$ is a vector $\mathbf{v}^*$ that satisfies $\mathbf{v}^* \geq \mathbf{v}_k$ for $k = 1, \ldots, K$ and for all $s \in S$, $v^*(s) = v_k(s)$ for at least one $k = 1, \ldots, K$. Let c-max denote the component-wise maximum, written as

$$\mathbf{v}^* := \text{c-max}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K\} \tag{5.5}$$

If vector $\mathbf{v}_{k^*}$ achieves the component-wise maximum, then

$$k^* \in \arg\underset{k=1,\ldots,K}{\text{c-max}}\{\mathbf{v}_k\} \tag{5.6}$$

Note that the component-wise maximum vector may not coincide with any of the vectors over which the component-wise maximum is taken, as the following example shows.

Example 5.2. Suppose $\mathbf{v}_1 = (2, 1)$, $\mathbf{v}_2 = (1, 3)$, and $\mathbf{v}_3 = (1, 5)$. Then $\mathbf{v}^* = \text{c-max}\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\} = (2, 5)$. Observe that $\mathbf{v}^* \notin \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$.

Where the component-wise maximum in a Markov decision process model is used, the situation in Example 5.2 cannot occur. In particular, c-max is used when taking the maximum of values defined over the set of all deterministic decision rules, where each decision rule is a vector of actions in $|S|$-dimensional space. Thus, the set of vectors over which the c-max is taken will correspond to the Cartesian product of the possible actions in all states, which will coincide with one of the decision rules.

Example 5.3. Consider a two-dimensional example where the first component can assume values 1 and 2 and the second component can assume values 3 and 5. The set of vectors generated by the Cartesian products of these component values is

$$A = \{(1, 3), (1, 5), (2, 3), (2, 5)\}.$$

The c-max of this set is (2,5), which is an element of A.

It is worth emphasizing that the component-wise maximum, c-max*, can be found by maximizing over each component separately, thus avoiding the enumeration of all possible combinations of component values.* This observation is especially important in Markov decision process computation, as shown later.

5.1.6 Norms

To establish the convergence of a sequence of iterates of a computational algorithm, a notion of distance in a vector space is required.

Definition 5.2. For $\mathbf{v} \in V$, define the *norm* of $\mathbf{v}$ by

$$\|\mathbf{v}\| := \max_{s \in S} |v(s)|. \tag{5.7}$$

This norm is often called the *sup-norm*[4]. It assigns as a "length" to $\mathbf{v}$ the absolute value of its largest component. For example, if $\mathbf{v} = (-2, 1)$, $\|\mathbf{v}\| = 2$. This norm applies equally well to non-finite dimensional state spaces where it is appropriate to replace "max" in (5.7) by a supremum (hence the name sup-norm). That degree of generality is not required here. There are many other norms (such as the Euclidean norm) that could be assigned to vectors in V, but in most of this book only the sup-norm is used.

The sup-norm on V induces a norm on $|S| \times |S|$ matrices $\mathbf{Q} : V \to V$ through the following definition.

Definition 5.3. For $\mathbf{Q} \in V \times V$, define the *matrix norm* of $\mathbf{Q}$ by

$$\|\mathbf{Q}\| := \sup_{\|\mathbf{v}\|=1} \|\mathbf{Q}\mathbf{v}\| = \max_{s \in S} \sum_{j \in S} |q(s, j)|. \tag{5.8}$$

The following easily proved result provides a useful inequality relating matrix and vector norms.

Lemma 5.1. For $\mathbf{v} \in V$ and matrix $\mathbf{Q} : V \to V$,

$$\|\mathbf{Q}\mathbf{v}\| \leq \|\mathbf{Q}\|\|\mathbf{v}\|. \tag{5.9}$$

The following definition combines the concepts of partial order and norm with a vector space.

Definition 5.4. A vector space V together with a partial order $\geq$ and a norm $\|\cdot\|$ is referred to as a *partially ordered normed linear space*.

This book (except Chapter 8) focuses on the vector space V of real-valued $|S|$-dimensional vectors with component-wise partial order and sup-norm.

[4]The sup-norm is often written as $\|\cdot\|_\infty$, to distinguish it from other norms such as the Euclidean norm, which is typically written as $\|\cdot\|_2$. For convenience, the subscript on the sup-norm is omitted.

5.2 The expected discounted reward of a policy

Definition 5.5. The *expected discounted reward* of policy $\pi \in \Pi^{\text{HR}}$ starting in state s with discount factor λ is[a]

$$v_\lambda^\pi(s) := \lim_{N\to\infty} E^\pi\left[\sum_{n=1}^{N} \lambda^{n-1} r(X_n, Y_n) \,\middle|\, X_1 = s\right]. \tag{5.10}$$

[a]Recall Y_n is a random variable that refers to the action chosen at decision epoch n, where it may be the result of a deterministic decision rule d applied to state X_n, or a randomized decision rule distributed according to $w_d(\cdot|X_n)$.

Below are two equivalent representations for $v_\lambda^\pi(s)$. Be sure to note the subtle distinctions between them. As a result of assuming bounded rewards and discount factor $0 \le \lambda < 1$, the bounded convergence theorem[5] justifies interchanging the limit and the expectation above to obtain

$$v_\lambda^\pi(s) = \lim_{N\to\infty} E^\pi\left[\sum_{n=1}^{N} \lambda^{n-1} r(X_n, Y_n) \,\middle|\, X_1 = s\right] = E^\pi\left[\sum_{n=1}^{\infty} \lambda^{n-1} r(X_n, Y_n) \,\middle|\, X_1 = s\right]. \tag{5.11}$$

Since the expectation of a sum equals the sum of expectations it follows again from the bounded convergence theorem that

$$v_\lambda^\pi(s) = \lim_{N\to\infty} \sum_{n=1}^{N} \lambda^{n-1} E^\pi\left[r(X_n, Y_n) \,|\, X_1 = s\right] = \sum_{n=1}^{\infty} \lambda^{n-1} E^\pi\left[r(X_n, Y_n) \,|\, X_1 = s\right]. \tag{5.12}$$

The representation on the right hand side of (5.12) will provide the basis for most of the following development.

5.2.1 Insights into how the expected discounted reward is evaluated

This section shows why the second representation for $v_\lambda^\pi(s)$ is so important, by looking into the expectations on the right hand side of (5.12) in depth. For any n,

$$E^\pi[r(X_n, Y_n) \,|\, X_1 = s] = \sum_{j\in S} \sum_{a\in A_j} r(j,a) p^\pi(X_n = j, Y_n = a | X_1 = s), \tag{5.13}$$

[5]The bounded convergence theorem states that if $|f_n(s)| \le W$ for all $s \in S$, $n \ge 1$ and some finite value W, then

$$\lim_{n\to\infty} \int_S f_n(s) ds = \int_S \lim_{n\to\infty} f_n(s) ds.$$

When S is finite sums replace integrals.

where p^π is a probability distribution incorporating both the transition probabilities and the action randomization distribution (see Section 2.2.3). Thus, combining (5.12) and (5.13),

$$v_\lambda^\pi(s) = \sum_{n=1}^{\infty} \sum_{j \in S} \sum_{a \in A_j} \lambda^{n-1} r(j,a) p^\pi(X_n = j, Y_n = a | X_1 = s). \tag{5.14}$$

As a consequence of Theorem 5.1, focus can be restricted to $\pi = (d_1, d_2, \ldots) \in \Pi^{\text{MR}}$ without loss of generality.

To gain insight into equation (5.14), the first three terms in the summation over n are written explicitly:

$$n = 1: \quad \sum_{a \in A_j} r(s,a) w_{d_1}(a|s) \tag{5.15}$$

$$n = 2: \quad \lambda \sum_{j \in S} \sum_{a' \in A_j} r(j,a') w_{d_2}(a'|j) \sum_{a \in A_s} p(j|s,a) w_{d_1}(a|s) \tag{5.16}$$

$$n = 3: \quad \lambda^2 \sum_{k \in S} \sum_{a'' \in A_j} r(k,a'') w_{d_3}(a''|k) \sum_{j \in S} \sum_{a' \in A_j} p(k|j,a') w_{d_2}(a'|j) \sum_{a \in A_s} p(j|s,a) w_{d_1}(a|s). \tag{5.17}$$

To derive the first term, note that states $j \neq s$ will have probability zero and $p^\pi(X_1 = s, Y_1 = a | X_1 = s) = p^\pi(Y_1 = a | X_1 = s) = w_{d_1}(a|s)$ by definition. Thus, the first term in the sum is simply the expected reward, where the expectation is with respect to the conditional probability of actions in decision epoch 1 given state s.

Using the law of total probability and the Markov property, the second term follows from

$$\begin{aligned} p^\pi(X_2 = j, Y_2 = a' | X_1 = s) &= p^\pi(Y_2 = a' | X_2 = j, X_1 = s) p^\pi(X_2 = j | X_1 = s) \\ &= w_{d_2}(a'|j) p^\pi(X_2 = j | X_1 = s) \\ &= w_{d_2}(a'|j) \sum_{a \in A_s} p^\pi(X_2 = j | X_1 = s, Y_1 = a) p^\pi(Y_1 = a | X_1 = s) \\ &= w_{d_2}(a'|j) \sum_{a \in A_s} p(j|s,a) w_{d_1}(a|s). \end{aligned}$$

Thus, (5.16) represents the expected discounted reward in decision epoch 2, where the expectation is with respect to the joint probability distribution of state and action at decision epoch 2 conditional on starting in state s at decision epoch 1. This probability distribution comprises both the distribution of actions that results from using the Markovian randomized decision rule d_2 in state j at decision epoch 2 and the distribution governing the transition from state s in epoch 1 to state j in epoch 2. Furthermore, the latter distribution is itself a product of the transition probabilities out of state s in epoch 1 for the possible actions in epoch 1 and the action randomization probabilities associated with decision rule d_1.

The third term (5.17) can be derived similarly and is left as an exercise. Its interpretation is similar to the previous term, but extended for one more decision epoch.

Simulating the discounted reward of a Markovian randomized policy

To provide another perspective on equation (5.10) (or equivalently (5.14)), consider the use of Monte Carlo simulation to estimate the value of a Markovian randomized policy $\pi = (d_1, d_2, \ldots)$. More details on how to implement it and other approaches appear in Chapter 10.

A challenge when simulating discounted rewards is to account for the infinite number of terms in (5.14). This can be achieved through truncation or using geometric stopping times. Putting that issue aside until Chapter 10, assume the goal is to obtain an estimate of the expected discounted reward after truncating it to K terms. The algorithm is expressed in terms of $r(s, a, j)$ and assumes knowledge of $p(j|s, a)$.

Algorithm 5.1. Simulating a single replicate of an estimate of $v_\lambda^\pi(s)$ for fixed $s \in S$.

1. **Initialize:** Specify $\pi = (d_1, d_2, \ldots) \in \Pi^{\text{MR}}$, $s \in S$ and $K \in \mathbb{Z}_+$. Set $k \leftarrow 1$, $v \leftarrow 0$.
2. **Iterate:** While $k \leq K$:
   (a) Sample a from $w_{d_k}(\cdot|s)$.
   (b) Sample s' from $p(\cdot|s, a)$.
   (c) $v \leftarrow v + \lambda^{k-1} r(s, a, s')$.
   (d) $s \leftarrow s'$.
   (e) $k \leftarrow k + 1$.
3. **Terminate:** Return v.

Note that if the reward was not a function of the next state steps 2(b) and 2(c) could be interchanged, with the update replaced by $v \leftarrow v + \lambda^{k-1} r(s, a)$. By generating many sample paths in this way, an estimate of the *distribution* of $v_\lambda^\pi(s)$ can be obtained.

5.2.2 Representing value functions as vectors

This section expresses quantities in the previous section using in vector/matrix notation. Doing simplifies explanations and derivations below.

The first three terms in the sum over n in equation (5.14) are

$$n = 1: \quad r_{d_1}(s) \tag{5.18}$$

$$n = 2: \quad \lambda \sum_{j \in S} r_{d_2}(j) P_{d_1}(j|s) \tag{5.19}$$

$$n = 3: \quad \lambda^2 \sum_{j \in S} r_{d_3}(j) P_{d_2}(j|k) P_{d_1}(k|s). \tag{5.20}$$

These expressions are the s-th components of $\mathbf{r}_{d_1}$, $\lambda \mathbf{P}_{d_1} \mathbf{r}_{d_2}$, and $\lambda^2 \mathbf{P}_{d_1} \mathbf{P}_{d_2} \mathbf{r}_{d_3}$, respectively. The last term can be written equivalently[6] as $\lambda^2 \mathbf{P}_\pi^2 \mathbf{r}_{d_3}$ using the two-step transition probability matrix $\mathbf{P}_\pi^2 = \mathbf{P}_{d_1} \mathbf{P}_{d_2}$.

Letting $\mathbf{P}_\pi^0 = \mathbf{I}$, the above pattern suggests that $\mathbf{v}_\lambda^\pi$ has the following closed form representation:

$$\mathbf{v}_\lambda^\pi = \sum_{n=1}^{\infty} \lambda^{n-1} \mathbf{P}_\pi^{n-1} \mathbf{r}_{d_n} \tag{5.21}$$

This infinite sum converges because $||\mathbf{P}_\pi^n|| = 1$, $0 \leq \lambda < 1$ and

$$\|\mathbf{v}_\lambda^\pi\| \leq \max_{a \in A_s, s \in S} |r(s,a)| (1-\lambda)^{-1}.$$

Expanding this sum and grouping terms appropriately, yields

$$\mathbf{v}_\lambda^\pi = \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \mathbf{r}_{d_2} + \lambda^2 \mathbf{P}_{d_1} \mathbf{P}_{d_2} \mathbf{r}_{d_3} + \cdots \tag{5.22}$$

$$= \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \left(\mathbf{r}_{d_2} + \lambda \mathbf{P}_{d_2} \mathbf{r}_{d_3} + \cdots \right). \tag{5.23}$$

Thus, if $\pi' = (d_2, d_3, \ldots)$ is a new policy that uses the same decision rules as π from decision epoch 2 onwards,

$$\mathbf{v}_\lambda^\pi = \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \mathbf{v}_\lambda^{\pi'}. \tag{5.24}$$

This equation can be interpreted as follows: the expected discounted reward associated with following policy π equals the immediate reward associated with using decision rule d_1 in the first epoch, plus the expected discounted reward over the infinite horizon, following policy π' from the second decision epoch onwards. Equivalently, this equation may be viewed as the expected value in a one-period problem, in which the terminal reward is the expected discounted reward associated with π'.

In component form, equation (5.24) can be written in general as

$$v_\lambda^\pi(s) = r_{d_1}(s) + \lambda \sum_{j \in S} P_{d_1}(j|s) v_\lambda^{\pi'}(j), \tag{5.25}$$

[6] Note the expression $\mathbf{P}_\pi^n$ denotes the n-step transition probability matrix corresponding to policy π. For stationary policies $\pi = d^\infty$, $\mathbf{P}_\pi^n$ equals the matrix power $\mathbf{P}_d^n$.

for all $s \in S$. If $\pi \in \Pi^{\text{MD}}$, then equation (5.24) becomes

$$v_\lambda^\pi(s) = r(s, d_1(s)) + \lambda \sum_{j \in S} p(j|s, d_1(s)) v_\lambda^{\pi'}(j) \tag{5.26}$$

for all $s \in S$. The subtle difference between (5.25) and (5.26) is that the former accommodates randomized decision rules (see equations (5.2) and (5.3)). Note that equation (5.26) is the discounted, infinite horizon generalization of equation (4.7) from the finite horizon setting with stationary rewards and transition probabilities.

Writing the above two expressions as expectations will be especially useful in Chapter 10. Regardless of whether π is deterministic or randomized

$$v_\lambda^\pi(s) = E^\pi[r(X_1, Y_1) + \lambda v_\lambda^{\pi'}(X_2)|X_1 = s]. \tag{5.27}$$

When the reward is a function of the next state as well, this expression becomes

$$v_\lambda^\pi(s) = E^\pi[r(X_1, Y_1, X_2) + \lambda v_\lambda^{\pi'}(X_2)|X_1 = s]. \tag{5.28}$$

5.2.3 Evaluating stationary policies

Stationary policies play an especially important role in discounted models. Restricting attention to such policies greatly simplifies the above expressions and more.

Let $d^\infty := (d, d, \ldots)$ denote the stationary policy that uses decision rule $d \in D^{\text{MR}}$ in every epoch. In this case, (5.21) becomes

$$\mathbf{v}_\lambda^{d^\infty} = \sum_{n=0}^{\infty} \lambda^n \mathbf{P}_d^n \mathbf{r}_d \tag{5.29}$$

and (5.24) becomes

$$\mathbf{v}_\lambda^{d^\infty} = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}_\lambda^{d^\infty}. \tag{5.30}$$

In other words, $\mathbf{v}_\lambda^{d^\infty}$ satisfies the system of linear equations. That it is the unique solution will now be established.

$$\mathbf{v} = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}. \tag{5.31}$$

A key lemma and its application

The following lemma, which is proved in the book Appendix (see Lemma B.1), is fundamental. It provides a matrix generalization of the representation for a convergent geometric series $\sum_{n=0}^{\infty} a^n = (1-a)^{-1}$ when $|a| < 1$.

Lemma 5.2. Let $\mathbf{Q}$ denote an $|S| \times |S|$ real-valued matrix for which $\mathbf{Q}^N \to \mathbf{0}$ as $N \to \infty$. Then the inverse of $\mathbf{I} - \mathbf{Q}$ exists and satisfies

$$(\mathbf{I} - \mathbf{Q})^{-1} = \sum_{n=0}^{\infty} \mathbf{Q}^n. \tag{5.32}$$

Applying it to (5.29) gives:

Theorem 5.2. Suppose $0 \leq \lambda < 1$ and $d \in D^{\mathrm{MR}}$. Then

$$\mathbf{v}_\lambda^{d^\infty} = (\mathbf{I} - \lambda \mathbf{P}_d)^{-1} \mathbf{r}_d \tag{5.33}$$

and is the unique solution of equation (5.31).

Proof. Let $d \in D^{\mathrm{MR}}$. Rewriting (5.31) yields

$$(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{r}_d. \tag{5.34}$$

As a result of Lemma 5.2, $\mathbf{I} - \lambda \mathbf{P}_d$ is invertible, since each row sum of $\lambda \mathbf{P}_d$ is strictly less than 1[7].

To establish uniqueness suppose $\mathbf{u}$ and $\mathbf{v}$ satisfy (5.34). Then $(\mathbf{I} - \lambda \mathbf{P}_d)(\mathbf{u} - \mathbf{v}) = \mathbf{0}$. That $\mathbf{v} = \mathbf{u}$ follows from the invertibility of $(\mathbf{I} - \lambda \mathbf{P}_d)$. □

The following example illustrates the computation of $\mathbf{v}_\lambda^{d^\infty}$ in two ways: direct solution of the system of linear equations (5.31) and using representation (5.33).

Example 5.4. Consider a discounted, infinite horizon version of the two-state model with a randomized stationary policy d^∞ that, in every epoch, chooses action $a_{1,1}$ with certainty in state s_1 and chooses state $a_{2,1}$ and $a_{2,2}$ with equal probability in state s_2.

Solving (5.31) directly: In component form, (5.31) becomes

$$v(s_1) = 5 \cdot 0.8 - 5 \cdot 0.2 + \lambda(0.8v(s_1) + 0.2v(s_2)) \tag{5.35}$$

$$v(s_2) = 0.5(-5 \cdot 1) + 0.5(20 \cdot 0.4 - 10 \cdot 0.6) + \lambda((0.5 \cdot 0 + 0.5 \cdot 0.4)v(s_1) + (0.5 \cdot 1 + 0.5 \cdot 0.6)v(s_2)), \tag{5.36}$$

[7] Alternatively, it suffices to show that the columns of $\mathbf{I} - \lambda \mathbf{P}_d$ are linearly independent. Suppose to the contrary that the columns are linearly dependent. That is, there exists $\mathbf{v} \neq \mathbf{0}$ such that $(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{0}$, which is equivalent to $\mathbf{v} = \lambda \mathbf{P}_d \mathbf{v}$. In other words, 1 is an eigenvalue of the matrix $\lambda \mathbf{P}_d$. From Theorem B.3 in the Appendix, since the largest eigenvalue of a transition probability matrix is 1, the largest eigenvalue of $\lambda \mathbf{P}_d$ is $\lambda < 1$, which is a contradiction.

which simplifies to

$$v(s_1) = 3 + \lambda(0.8v(s_1) + 0.2v(s_2)) \tag{5.37}$$
$$v(s_2) = -1.5 + \lambda(0.2v(s_1) + 0.8v(s_2)). \tag{5.38}$$

Solving for $v(s_1)$ and $v(s_2)$,

$$v_\lambda^{d^\infty}(s_1) = \frac{3(1-2.7\lambda)}{(1-\lambda)(1-0.6\lambda)}, \qquad v_\lambda^{d^\infty}(s_2) = \frac{3(0.6\lambda-0.5)}{(1-\lambda)(1-0.6\lambda)}. \tag{5.39}$$

Using (5.33): From (5.33):

$$\mathbf{I} - \lambda\mathbf{P}_d = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \lambda \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} = \begin{pmatrix} 1-0.8\lambda & -0.2\lambda \\ -0.2\lambda & 1-0.8\lambda \end{pmatrix} \tag{5.40}$$

Then

$$(\mathbf{I} - \lambda\mathbf{P}_d)^{-1}\mathbf{r}_d = \frac{1}{(1-\lambda)(1-0.6\lambda)} \begin{pmatrix} 1-0.8\lambda & 0.2\lambda \\ 0.2\lambda & 1-0.8\lambda \end{pmatrix} \begin{pmatrix} 3 \\ -1.5 \end{pmatrix}, \tag{5.41}$$

which can easily be verified to be the same as (5.39).

Next, the expected discounted reward for the four deterministic stationary policies are computed using (5.33). Setting $\lambda = 0.9$, $d_1 = (a_{1,1}, a_{2,1})$, $d_2 = (a_{1,1}, a_{2,2})$, $d_3 = (a_{1,2}, a_{2,1})$ and $d_4 = (a_{1,2}, a_{2,2})$ yields

$$\mathbf{v}_\lambda^{d_1^\infty} = \begin{bmatrix} -21.429 \\ -50 \end{bmatrix}, \mathbf{v}_\lambda^{d_2^\infty} = \begin{bmatrix} 27.188 \\ 25.625 \end{bmatrix}, \mathbf{v}_\lambda^{d_3^\infty} = \begin{bmatrix} -40 \\ -50 \end{bmatrix} \text{ and } \mathbf{v}_\lambda^{d_4^\infty} = \begin{bmatrix} 30.147 \\ 27.941 \end{bmatrix}.$$

Observe that $\mathbf{v}_\lambda^{d_4^\infty} \geq \mathbf{v}_\lambda^{d_2^\infty} \geq \mathbf{v}_\lambda^{d_1^\infty} \geq \mathbf{v}_\lambda^{d_3^\infty}$ so that the stationary policy d_4^∞ is optimal within the class of stationary policies. This implies d_4^∞ is optimal, once it is established that stationary policies are optimal within the class of all policies, which is done below.

Properties of $(\mathbf{I} - \lambda\mathbf{P}_d)^{-1}$

From Lemma 5.2 it follows immediately that:

Theorem 5.3. Suppose $0 \leq \lambda < 1$ and $d \in D^{\text{MR}}$, then

$$(\mathbf{I} - \lambda\mathbf{P}_d)^{-1} = \sum_{n=0}^{\infty} \lambda^n \mathbf{P}_d^n. \tag{5.42}$$

Several properties of $(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}$ appear in the following lemma.

Lemma 5.3. Suppose $0 \leq \lambda < 1$ and $\mathbf{u}, \mathbf{v} \in V$. Then, for any $d \in D^{\mathrm{MR}}$:

1. If $\mathbf{u} \geq \mathbf{0}$, then $(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{u} \geq \mathbf{u} \geq \mathbf{0}$.
2. If $\mathbf{u} \geq \mathbf{v}$, then $(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{u} \geq (\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{v}$.
3. $(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{e} = (1 - \lambda)^{-1}\mathbf{e}$.

Proof. From Theorem 5.3,

$$(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{u} = \mathbf{u} + \lambda \mathbf{P}_d \mathbf{u} + \lambda^2 \mathbf{P}_d^2 \mathbf{u} + \ldots \geq \mathbf{u} \geq \mathbf{0}$$

since $\lambda \geq 0$ and all entries of $\mathbf{P}_d$ are non-negative. The second result follows by applying the first result to $\mathbf{u} - \mathbf{v}$. The third result follows from (5.42), the fact that $\mathbf{P}_d \mathbf{e} = \mathbf{e}$ and the formula for the sum of a geometric series. □

5.3 Optimal policies and the Bellman equation

This section defines optimal policies for discounted models, introduces the Bellman (optimality) equation and shows that the existence of a solution to the Bellman equation implies the existence of an optimal policy that is stationary.

5.3.1 Optimal policies

Definitions of optimal policies and optimal value functions for infinite horizon discounted models follow.

Definition 5.6. An *optimal policy* $\pi^* \in \Pi^{\mathrm{HR}}$ satisfies

$$v_\lambda^{\pi^*}(s) \geq v_\lambda^\pi(s) \tag{5.43}$$

for all $\pi \in \Pi^{\mathrm{HR}}$ and $s \in S$.

Definition 5.7. The *optimal value function* $v_\lambda^*(s)$ is defined by

$$v_\lambda^*(s) := \sup_{\pi \in \Pi^{\mathrm{HR}}} v_\lambda^\pi(s) \tag{5.44}$$

for all $s \in S$.

Definition 5.8. For any $\epsilon > 0$, an *ϵ-optimal policy* $\pi^\epsilon \in \Pi^{\text{HR}}$ satisfies

$$v_\lambda^{\pi^\epsilon}(s) \geq v_\lambda^*(s) - \epsilon \tag{5.45}$$

for all $s \in S$.

These definitions are similar to those in finite horizon models under the expected total reward criterion. Recall that in finite horizon models with finite action sets, Markovian deterministic policies are optimal within the class of history-dependent randomized policies. In infinite horizon discounted models with stationary rewards and transition probabilities, a stronger result holds, namely that *stationary deterministic policies are optimal within the class of history-dependent randomized policies*. Theorem 5.4 provides the main result of this section. The remainder of the section develops the necessary machinery to establish it. The assumption of finite action sets is added for emphasis.

Theorem 5.4. Suppose A_s is finite for each $s \in S$. Then there exists $\pi^* \in \Pi^{\text{SD}}$ for which

$$v_\lambda^{\pi^*}(s) = v_\lambda^*(s)$$

for all $s \in S$.

The main consequence of this theorem is that the search for optimal policies can be restricted to the class of stationary deterministic policies[8].

5.3.2 The Bellman equation for a discounted model

Chapter 4 showed that in a finite horizon model, the following sequence of recursive equations together with a terminal condition (re-expressed in the notation of this chapter) could be used to determine optimal value functions and policies:

$$v_n(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_{n+1}(j)) \right\}. \tag{5.46}$$

Informally, passing to the limit on both sides of (5.46) suggests that the Bellman equation for a discounted model should be of the form:

[8]Note that stationary optimal policy and optimal stationary policy are two different concepts. The former refers to the fact that the optimal policy is stationary. The latter refers to the best stationary policy among all stationary policies, but it may not be optimal among all policies, including those that are non-stationary. The focus here is on the former.

$$v(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}. \tag{5.47}$$

The following development will show that this heuristic derivation is justified by showing that the Bellman equation has a unique solution and that the solution equals the optimal value function.

Vector form of the Bellman equation and c-max notation

Equation (5.47) can be expressed in vector notation as:

$$\mathbf{v} = \underset{d \in D^{\text{MD}}}{\text{c-max}} \left\{ \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} \right\}. \tag{5.48}$$

It is important to observe that the vector Bellman equation uses c-max instead of max[9]. The use of the expression c-max on the right-hand side of (5.48) emphasizes that the maximum is taken component-wise. This means that given a $\mathbf{v} \in V$, for each $s \in S$ one evaluates

$$r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \tag{5.49}$$

for each $a \in A_s$, and then sets the right hand side of (5.48) to the vector of values that achieve this maximum. Example 5.2 expands on this point and emphasizes that the maximum can be taken component by component, thus avoiding enumerating $|A_{s_1}| \times |A_{s_2}| \times \ldots \times |A_{s_M}|$ Markovian deterministic decision rules.

The expression

$$d^* \in \arg \underset{d \in D^{\text{MD}}}{\text{c-max}} \left\{ \mathbf{r}_d + \lambda \mathbf{P} \mathbf{v} \right\} \tag{5.50}$$

[9]The c-max notation is not standard, but in our teaching experience, using max instead was a source of confusion among students. When the equation was written with max instead of c-max, it was frequently interpreted incorrectly to represent the process of:

1. Enumerating all decision rules.
2. Evaluating $\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}$ for each decision rule d.
3. Setting $\max_{d \in D^{\text{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}$ to the maximum value obtained in step 2.

Such an approach runs contrary to the dynamic programming principle of reducing a complex problem into a series of simpler problems that can be analyzed state by state.

indicates that the decision rule d^* is obtained by choosing

$$d^*(s) \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}$$

for each $s \in S$.

5.3.3 The maximum return operator

Definition 5.9. The *maximum return operator* or *Bellman operator* $L : V \to V$ is defined as

$$L\mathbf{v} := \text{c-}\max_{d \in D^{\text{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}. \tag{5.51}$$

This is a nonlinear operator[10] that is applied to a vector $\mathbf{v} \in V$ and returns the vector of values associated with the deterministic decision rule that obtains the component-wise maximum of $\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}$. Note that for $\mathbf{v} \in V$, $L\mathbf{v}$ is a vector in V and its s-th component is represented by $L\mathbf{v}(s)$. Using L simplifies notation, makes proofs more transparent and suggests analyses for other optimality criteria.

The following lemma is a direct application of Lemma 2.1 expressed in vector notation. It asserts that the component-wise maximum over Markovian deterministic decision rules equals the component-wise maximum over Markovian randomized decision rules. Note it doesn't require that λ be less than 1.

Lemma 5.4. For any $\mathbf{v} \in V$ and $0 \leq \lambda \leq 1$

$$\text{c-}\max_{d \in D^{\text{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\} = \text{c-}\max_{d \in D^{\text{MR}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}. \tag{5.52}$$

Consequently (5.48) can be expressed[11] as

Bellman equation: operator notation

$$\mathbf{v} = L\mathbf{v}. \tag{5.53}$$

A solution of equation (5.53) is called a *fixed point* of L. Theoretical results regarding existence of a fixed point for a contraction mapping, which is defined below, are used to prove that the Bellman equation has a unique solution.

[10] In a finite state and action model, L is *piecewise linear* and concave.

[11] This representation indicates why this book typically refers to the Bellman equation in the singular and not "Bellman equations". However, in some instances, when the corresponding system of equations is written in component notation, they may be referred to as the Bellman equations.

Define the operator $L_d : V \to V$ for $d \in D^{\text{MR}}$ as

$$L_d\mathbf{v} := \mathbf{r}_d + \lambda \mathbf{P}_d\mathbf{v}. \tag{5.54}$$

That is, L_d gives the value of using decision rule d for one period and then receiving a terminal reward $\mathbf{v}$. Note that the fixed point equation $L_d\mathbf{v} = \mathbf{v}$ previously appeared as (5.31) where it was shown that $\mathbf{v}_\lambda^{d^\infty}$ is the unique fixed point of L_d when $0 \leq \lambda < 1$.

The expression $L^2\mathbf{v}$ corresponds to applying L to $\mathbf{v}$ twice[12]. Since $L\mathbf{v} \in V$, it is in the domain of L, so that the operator L can be applied to it yielding $L^2\mathbf{v} = L(L\mathbf{v})$. In general $L^n\mathbf{v} = L(L^{n-1}\mathbf{v})$.

Contraction mappings

Contraction mappings are an important analytic concept that is fundamental to establishing many key results in this chapter and Chapter 6.

Definition 5.10. An operator $T : V \to V$ is a *contraction mapping* on V with *modulus* κ, if for all $\mathbf{u}$ and $\mathbf{v}$ in V there exists a scalar $0 \leq k < 1$ for which

$$\|T\mathbf{v} - T\mathbf{u}\| \leq \kappa\|\mathbf{v} - \mathbf{u}\|. \tag{5.55}$$

It is easy to show that for any $n \geq 1$,

$$\|T^n\mathbf{v} - T^n\mathbf{u}\| \leq \kappa^n\|\mathbf{v} - \mathbf{u}\|.$$

A normed linear space is said to be *complete* if whenever $\|\mathbf{v}^n - \mathbf{v}^m\| \to 0$ as $m \to \infty$ and $n \to \infty$, there exists a $\mathbf{v}^* \in V$ for which $\|\mathbf{v}^n - \mathbf{v}^*\| \to 0$[13]. Note that the set of real numbers is complete, while the set of rational numbers is not. In the context of this book, the set of all $|S|$-dimensional real vectors, V, is complete.

The following result, often referred to as the *Banach fixed point theorem* or the *contraction mapping fixed point theorem* will be the basis for establishing a unique solution of the Bellman equation expressed as $L\mathbf{v} = \mathbf{v}$.

[12] It does not mean squaring $L\mathbf{v}$ which would be written as $(L\mathbf{v})^2$.

[13] Equivalently, a normed linear space is complete if every Cauchy sequence converges to an element of that space. A complete normed linear space is called a *Banach* space.

Theorem 5.5. Banach Fixed Point Theorem
Let V be a complete normed linear space and $T : V \to V$ be a contraction mapping. Then:

1. there exists a unique $\mathbf{v}^* \in V$ that satisfies $T\mathbf{v}^* = \mathbf{v}^*$, and
2. for any $\mathbf{v}^0 \in V$, the sequence $\{\mathbf{v}^n, n = 0, 1, 2, \ldots\}$, defined by

$$\mathbf{v}^{n+1} = T\mathbf{v}^n = T^{n+1}\mathbf{v}^0 \tag{5.56}$$

converges to $\mathbf{v}^*$. That is, $\|\mathbf{v}^n - \mathbf{v}^*\| \to 0$ as $n \to \infty$.

Proofs of this result are widely available and surprisingly straightforward. This result is important because when T is a contraction mapping, then the first part of Theorem 5.5 establishes the existence of a unique fixed point for T in V. The second part establishes that repeated application of the recursion $\mathbf{v} \leftarrow T\mathbf{v}$, converges to a fixed point of T.

Proposition 5.1. Suppose $0 \leq \lambda < 1$. Then the operator L defined in (5.51) is a contraction mapping on V with modulus λ.

Proof. Let[14] $\mathbf{u}$ and $\mathbf{v}$ be elements of V. Fix $s \in S$ and assume without loss of generality that $L\mathbf{v}(s) \geq L\mathbf{u}(s)$. Then by the finiteness of A_s, there exists an $a_s^* \in A_s$ for which

$$r(s, a_s^*) + \lambda \sum_{j \in S} p(j|s, a_s^*)v(j) = \max_{a \in A_s} \left\{ r(s, a) + \lambda \sum_{j \in S} p(j|s, a)v(j) \right\} = L\mathbf{v}(s). \tag{5.57}$$

Since a_s^* need not be a maximizer of $L\mathbf{u}(s)$,

$$r(s, a_s^*) + \lambda \sum_{j \in S} p(j|s, a_s^*)u(j) \leq \max_{a \in A_s} \left\{ r(s, a) + \lambda \sum_{j \in S} p(j|s, a)u(j) \right\} = L\mathbf{u}(s). \tag{5.58}$$

Subtracting (5.58) from (5.57) and noting the assumption that $L\mathbf{v}(s) \geq L\mathbf{u}(s)$, it follows that

$$\begin{aligned} 0 &\leq L\mathbf{v}(s) - L\mathbf{u}(s) \\ &\leq r(s, a_s^*) + \lambda \sum_{j \in S} p(j|s, a_s^*)v(j) - \left(r(s, a_s^*) + \lambda \sum_{j \in S} p(j|s, a_s^*)u(j) \right) \\ &= \lambda \sum_{j \in S} p(j|s, a_s^*)\Big(v(j) - u(j)\Big) \leq \lambda \sum_{j \in S} p(j|s, a_s^*)\|\mathbf{v} - \mathbf{u}\| \end{aligned}$$

[14] The proof uses component notation because two vectors need not be comparable. Note also for scalars, that $x \leq y$ does not imply $|x| \leq |y|$. It requires $0 \leq x \leq y$.

$$= \lambda \|\mathbf{v} - \mathbf{u}\|.$$

Since s was arbitrary and a similar result holds for $L\mathbf{u}(s) - L\mathbf{v}(s)$ assuming $L\mathbf{u}(s) \geq L\mathbf{v}(s)$, it follows that

$$\|L\mathbf{v} - L\mathbf{u}\| \leq \lambda \|\mathbf{v} - \mathbf{u}\|$$

completing the proof. □

Combining the above results and noting that the set of real-valued n-dimensional vectors V is complete leads to the following important result.

Theorem 5.6. Suppose $0 \leq \lambda < 1$. Then $L\mathbf{v} = \mathbf{v}$ has a unique solution in V.

Note that by replacing D^{MD} with a single decision rule[15] $d \in D^{\text{MD}}$, this result also establishes that $\mathbf{v}_\lambda^{d^\infty}$ is the unique fixed point of $L_d\mathbf{v} = \mathbf{v}$ in V. As a result of Lemma 5.4 it generalizes to $d \in D^{\text{MR}}$ to give the following result that previously appeared as Theorem 5.2 and was proved without appealing to the Banach fixed point theorem.

Corollary 5.1. For each $d \in D^{\text{MR}}$, $\mathbf{v}_\lambda^{d^\infty}$ is the unique fixed point of $L_d\mathbf{v} = \mathbf{v}$ in V.

5.3.4 Existence of an optimal value function

That the solution of the Bellman equation is the optimal value function follows from the following important theorem, which also can be used to derive upper and lower bounds on the optimal value function.

Theorem 5.7. Let $0 \leq \lambda < 1$. If there exists a $\mathbf{v} \in V$ for which:

1. $\mathbf{v} \geq L\mathbf{v}$, then $\mathbf{v} \geq \mathbf{v}_\lambda^*$,
2. $\mathbf{v} \leq L\mathbf{v}$, then $\mathbf{v} \leq \mathbf{v}_\lambda^*$ an
3. $\mathbf{v} = L\mathbf{v}$, then $\mathbf{v} = \mathbf{v}_\lambda^*$. Moreover, it is the unique solution of this equation.

Proof. To prove the first result, choose an arbitrary policy $\pi = (d_1, d_2, \ldots) \in \Pi^{\text{MR}}$ and an $\epsilon > 0$.

Combining $L\mathbf{v} \geq \mathbf{v}$ with Lemma 5.4, it follows that

$$\mathbf{v} \geq L\mathbf{v} = \underset{d \in D^{\text{MD}}}{\text{c-max}}\{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\} = \underset{d \in D^{\text{MR}}}{\text{c-max}}\{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\} \geq \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}$$

[15] This "trick" will be used on occasion in the sequel to obtain results for individual policies by first establishing results in terms of the maximum return operator.

for all $d \in D^{\mathrm{MR}}$. Hence

$$\mathbf{v} \geq \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1}\mathbf{v} \geq \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1}(\mathbf{r}_{d_2} + \lambda \mathbf{P}_{d_2}\mathbf{v}) = \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1}\mathbf{r}_{d_2} + \lambda^2 \mathbf{P}_{d_1}\mathbf{P}_{d_2}\mathbf{v}$$

Repeating this argument, it follows that for $n \geq 1$,

$$\begin{aligned}\mathbf{v} &\geq \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1}\mathbf{r}_{d_2} + \cdots + \lambda^{n-1}\mathbf{P}_{\pi}^{n-1}\mathbf{r}_{d_n} + \lambda^n \mathbf{P}_{\pi}^n \mathbf{v} \\ &= \mathbf{v}_{\lambda}^{\pi} - \sum_{k=n}^{\infty} \lambda^k \mathbf{P}_{\pi}^k \mathbf{r}_{d_{k+1}} + \lambda^n \mathbf{P}_{\pi}^n \mathbf{v}\end{aligned}$$

where the last equality follows from the definition of $\mathbf{v}_{\lambda}^{\pi}$ in (5.21). Choosing n sufficiently large, ensures that each of the last two terms can be bounded above in norm by $\epsilon/2$. This establishes that

$$\mathbf{v} \geq \mathbf{v}_{\lambda}^{\pi} - \epsilon \mathbf{e}$$

for any $\epsilon > 0$ and $\pi \in \Pi^{\mathrm{MR}}$.

From this inequality and Theorem 5.1 it follows that

$$\mathbf{v} \geq \sup_{\pi \in \Pi^{\mathrm{MR}}} \mathbf{v}_{\lambda}^{\pi} = \sup_{\pi \in \Pi^{\mathrm{HR}}} \mathbf{v}_{\lambda}^{\pi} = \mathbf{v}_{\lambda}^{*}$$

which proves part 1.

To prove the second result, since

$$\mathbf{v} \leq L\mathbf{v} = \underset{d \in D^{\mathrm{MD}}}{\text{c-max}}\{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\},$$

there exists a specific $d \in D^{\mathrm{MD}}$ for which $\mathbf{v} \leq \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}$. Hence

$$\mathbf{v} \leq \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} \leq \mathbf{r}_d + \lambda \mathbf{P}_d(\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}) = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{r}_d + \lambda^2 \mathbf{P}_d^2 \mathbf{v}$$

so by repeating this argument, it follows that

$$\mathbf{v} \leq \sum_{n=1}^{\infty} \lambda^{n-1} \mathbf{P}_d^{n-1} \mathbf{r}_d = \mathbf{v}_{\lambda}^{d^\infty} \leq \sup_{\pi \in \Pi^{\mathrm{HR}}} \mathbf{v}_{\lambda}^{\pi},$$

as desired.

Combining parts 1 and 2 establishes that if $\mathbf{v} = L\mathbf{v}$, $\mathbf{v} = \mathbf{v}_{\lambda}^{*}$. □

Combining Theorem 5.7 with Theorem 5.6 yields the following fundamental result.

Theorem 5.8. The optimal value function $\mathbf{v}_{\lambda}^{*}$ is the unique solution to the optimality equation

$$\mathbf{v} = L\mathbf{v} = \underset{d \in D^{\mathrm{MD}}}{\text{c-max}}\{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}. \tag{5.59}$$

A version of Theorem 5.7 in component notation follows. It will be especially useful for formulating a discounted Markov decision process as a linear program in Section 5.9.

Corollary 5.2. Let $\mathbf{v} \in V$.

1. If
$$v(s) \geq \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}$$
for all $s \in S$, then $v(s) \geq v^*_\lambda(s)$ for all $s \in S$, and

2. If
$$v(s) \leq \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}$$
for all $s \in S$, then $v(s) \leq v^*_\lambda(s)$ for all $s \in S$.

3. The optimal value function $v^*_\lambda(s)$ is the unique solution of
$$v(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}.$$

5.3.5 Existence of an optimal stationary policy

This section shows how to find an optimal policy from a solution of the Bellman equation $\mathbf{v}^*_\lambda$. The following is the fundamental result.

Theorem 5.9. Suppose there exists a $d^* \in D^{\mathrm{MD}}$ satisfying
$$d^* \in \arg \operatorname*{c\text{-}max}_{d \in D^{\mathrm{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}^*_\lambda\}. \tag{5.60}$$
Then $(d^*)^\infty$ is an optimal deterministic stationary policy.

Proof. Consider any decision rule $d^* \in D^{\mathrm{MD}}$ satisfying (5.60). Then
$$\mathbf{v}^*_\lambda = L\mathbf{v}^*_\lambda = \operatorname*{c\text{-}max}_{d \in D^{\mathrm{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}^*_\lambda\} = \mathbf{r}_{d^*} + \lambda \mathbf{P}_{d^*} \mathbf{v}^*_\lambda = L_{d^*} \mathbf{v}^*_\lambda$$
But since $\mathbf{v}_\lambda^{(d^*)^\infty}$ is the unique solution of $L_{d^*}\mathbf{v} = \mathbf{v}$, it follows from Corollary 5.1 that $\mathbf{v}_\lambda^{(d^*)^\infty} = \mathbf{v}^*_\lambda$. Hence $(d^*)^\infty$ is optimal. □

Note that this result shows that a stationary policy derived from *any* d^* satisfying (5.60) is optimal. Moreover, if more than one policy is optimal, any policy that ran-

domizes between actions chosen by different deterministic policies is optimal. This is particularly relevant because some policies with the same expected discounted reward might have smaller variance.

What is required to ensure that such a d^* exists? Clearly when A_s is finite for each $s \in S$ this holds, but it holds in greater generality under continuity and compactness assumptions.

A restatement of this result in component notation follows.

Corollary 5.3. Suppose for all $s \in S$ there exists an a_s^* satisfying

$$a_s^* \in \arg\max_{a \in A_s} \left\{ r(s,a) + \sum_{j \in S} \lambda p(j|s,a) v_\lambda^*(j) \right\} \tag{5.61}$$

and let $d^*(s) = a_s^*$ for all $s \in S$. Then $(d^*)^\infty$ is an optimal policy.

The following terminology will be useful.

Definition 5.11. A stationary policy $(d^*)^\infty$ where d^* satisfies (5.60) is said to be a *greedy policy*. A decision rule satisfying (5.60) is said to be a *greedy decision rule* and an action satisfying (5.61) is said to be a *greedy action*. Decision rules and actions selected in this way are said to be chosen *greedily*.

Two examples

Example 5.5. This example revisits the two-state model. After simplification, the optimality equations in component form are given by

$$v(s_1) = \max\{3 + \lambda(0.8v(s_1) + 0.2v(s_2)),\ 5 + \lambda v(s_2)\} \tag{5.62}$$
$$v(s_2) = \max\{-5 + \lambda v(s_2),\ 2 + \lambda(0.4v(s_1) + 0.6v(s_2))\}. \tag{5.63}$$

Direct computation in Example 5.4 based on enumerating all stationary policies showed that when $\lambda = 0.9$, the stationary policy d_4^∞, where $d_4 = (a_{1,2}, a_{2,2})$, is optimal and the optimal value function is $\mathbf{v}_\lambda^{d_4^\infty} = (30.147, 27.941)$. It can be easily verified that this value function satisfies the Bellman equation. In fact, d_4^∞ is the optimal stationary policy for all λ, which is left as an exercise to show.

The following example illustrates an algebraic solution of the Bellman equation. Because of its simple structure, solving it does not require enumeration or knowledge of the optimal value function *a priori*. In contrast to the previous example, it also shows that an optimal policy may depend on the specific value of λ.

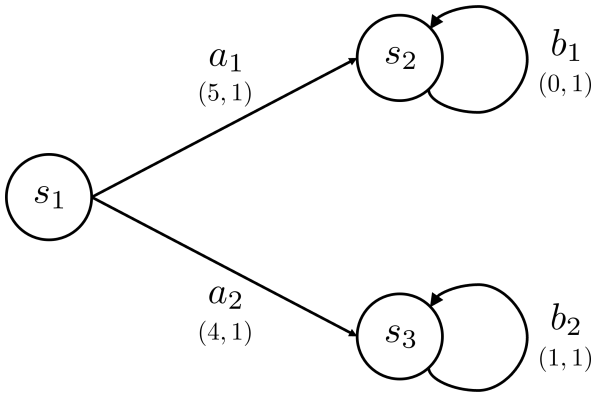


Figure 5.2: The three-state Markov decision process from Example 5.6 to illustrate the effect of varying λ on the optimal policy.

Example 5.6. Consider a three-state Markov decision process with $S = \{s_1, s_2, s_3\}$, $A_{s_1} = \{a_1, a_2\}$, $A_{s_2} = \{b_1\}$ and $A_{s_3} = \{b_2\}$, shown in Figure 5.2. Under action a_1, the process generates a reward of 5, transitions to state s_2 with probability 1 and then remains at state s_2 forever. Subsequent self-transitions in state s_2, (under action b_1), generate a reward of 0. Under action a_2, the process generates a reward of 4, transitions to state s_3 with probability 1, and then remains at state s_3 forever. Subsequent self-transitions in state s_3, (under action b_2), generate a reward of 1. The horizon is infinite and the discount factor is λ.

The Bellman equation for this model is:

$$\begin{aligned} v(s_1) &= \max\{5 + \lambda v(s_2), 4 + \lambda v(s_3)\} \\ v(s_2) &= 0 + \lambda v(s_2) \\ v(s_3) &= 1 + \lambda v(s_3) \end{aligned}$$

Because of the simple structure, these equations can be solved directly to obtain

$$v^*_\lambda(s) = \begin{cases} \max\{5, 4 + \lambda/(1-\lambda)\} & s = s_1 \\ 0 & s = s_2, \\ 1/(1-\lambda) & s = s_3, \end{cases}$$

The solution depends on λ as follows:

$$v^*_\lambda(s_1) = \begin{cases} 5 & 0 \le \lambda \le 0.5 \\ 4 + \lambda/(1-\lambda) & 0.5 \le \lambda < 1 \end{cases}$$

Thus, from (5.60) the stationary policy that uses action a_1 in state s_1 is optimal when $0 \le \lambda \le 0.5$ and the stationary policy that uses action a_2 in state s_1 is

optimal when $0.5 \leq \lambda < 1$. Note that both actions are optimal at $\lambda = 0.5$, which means that a randomized policy is also optimal at $\lambda = 0.5$.

Intuitively, small λ values mean that future rewards are heavily discounted relative to immediate rewards, so the decision maker would prefer to take the larger immediate reward of 5, and forego all future rewards. On the other hand, a large λ means that it is better to take the smaller initial reward of 4, but be assured of a future, infinite stream of rewards of 1, even though they are discounted.

Demonstrating that there exists an optimal stationary policy in Theorem 5.9 required first finding $\mathbf{v}_\lambda^*$. While an enumeration approach in Example 5.5 may be feasible for small "toy" problems, it is not scalable to larger problems. Thus, subsequent sections present algorithms to compute $\mathbf{v}_\lambda^*$ and corresponding optimal stationary policies.

5.4 State-action value functions

Section 4.5 introduced the concept of a state-action value function. This concept will not be central to this chapter, it is include for ease of reference. They will be fundamental in Chapters 10 and 11.

For a discounted model, define the *state-action value function corresponding to policy* π for $\pi \in \Pi^{\text{HR}}$, denoted by $q_\lambda^\pi(s,a)$, to be the infinite horizon expected discounted reward when choosing action $a \in A_s$ in state $s \in S$ at the current decision epoch and using policy π subsequently. Consequently,

$$q_\lambda^\pi(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_\lambda^\pi(j) \tag{5.64}$$

Similarly, define *the optimal state-action value function*, denoted by $q_\lambda^*(s,a)$, to be the infinite horizon expected discounted reward when action $a \in A_s$ is chosen in state $s \in S$ at the current decision epoch and the optimal policy is used subsequently. It satisfies

$$q_\lambda^*(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_\lambda^*(j) \tag{5.65}$$

for all $a \in A_s$ and $s \in S$. When an optimal policy π^* exists, (5.65) is equivalent to (5.64) with $\pi = \pi^*$.

When either $v_\lambda^\pi(s)$ or $v_\lambda^*(s)$ are available the above two representations can be used to evaluate the state-action value functions. However, from the perspective of their intended use, recursions expressed directly in terms of either $q_\lambda^\pi(s,a)$ or $q_\lambda^*(s,a)$ are preferable.

5.4.1 State-action value function recursions for a fixed policy

When $\pi = d^\infty$ for some $d \in D^{\mathrm{MD}}$,

$$v_\lambda^{d^\infty}(s) = q_\lambda^{d^\infty}(s, d(s)).$$

Hence from (5.64)

$$q_\lambda^{d^\infty}(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) q_\lambda^{d^\infty}(j, d(j)). \tag{5.66}$$

When the reward depends on the subsequent state, (5.66), can be expressed as

$$q_\lambda^{d^\infty}(s,a) = \sum_{j \in S} p(j|s,a) \Big(r(s,a,j) + \lambda q_\lambda^{d^\infty}(j, d(j)) \Big). \tag{5.67}$$

When d^∞ is randomized, the state action value function satisfies

$$q_\lambda^{d^\infty}(s,a) = \sum_{j \in S} p(j|s,a) \Big(r(s,a,j) + \lambda \sum_{a' \in A_j} w_d(a'|s) q_\lambda^{d^\infty}(j,a') \Big). \tag{5.68}$$

Observe that in contrast to the standard value function recursion

$$v_\lambda^{d^\infty} = r(s, d(s)) + \lambda \sum_{j \in S} p(j|s, d(s)) v_\lambda^{d^\infty}(j)$$

the decision rule $d(s)$ in (5.66) only appears in the q-function, and not the reward function or transition probabilities.

5.4.2 Bellman equation for state-action value functions

It follows from the Bellman equation (5.47) that

$$v_\lambda^*(s) = \max_{a \in A_s} q_\lambda^*(s,a).$$

Consequently, the optimal state-action value function may be represented as follows.

Bellman equation for state-action value functions
For $s \in S$ and $a \in A_s$,

$$q_\lambda^*(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) \max_{a' \in A_s} q_\lambda^*(j,a') \tag{5.69}$$

or, when the reward depends on the subsequent state,

$$q_\lambda^*(s,a) = \sum_{j \in S} p(j|s,a) \left(r(s,a,j) + \lambda \max_{a' \in A_s} q_\lambda^*(j,a') \right). \tag{5.70}$$

Choosing $d^*(s) \in \arg\max_{a \in A_s} q^*(s,a)$ produces an optimal stationary policy.

Written as an expected value, (5.69) is equivalent to

$$q^*(s,a) = E^{d^*}\left[r(X,a) + \lambda \max_{a' \in A_{X'}} q^*(X',a') \,\middle|\, X = s, Y = a \right], \tag{5.71}$$

where $(d^*)^\infty$ is an optimal policy for the discounted model and X' is distributed according to $p(\cdot|s,a)$. When $r(s,a,j)$ is the model primitive, (5.70) becomes:

$$q^*(s,a) = E^{d^*}\left[r(X,Y,X') + \lambda \max_{a' \in A_{X'}} q^*(X',a') \,\middle|\, X = s, Y = a \right]. \tag{5.72}$$

Observe that these two expressions differ from those representing the Bellman equation based on the value function. Here, the maximization is *inside* the expectation. While this difference is unimportant in this chapter, it will have a significant impact in Chapters 10 and 11, where expectations are estimated by random draws from a transition probability distribution or samples from the environment. As before, (5.66) may be viewed as special case of (5.69) when $A_s = \{d(s)\}$.

5.4.3 An example

This example derives the Bellman equation in component form for the two-state model using representation (5.70). For $s = s_1$,

$$\begin{aligned}
q(s_1,a_{1,1}) &= p(s_1|s_1,a_{1,1})\left(r(s_1,a_{1,1},s_1) + \lambda \max_{a \in A_{s_1}} q(s_1,a) \right) \\
&\quad + p(s_2|s_1,a_{1,1})\left(r(s_1,a_{1,1},s_2) + \lambda \max_{a \in A_{s_2}} q(s_2,a) \right) \\
&= 0.8\left(5 + \lambda \max_{a \in A_{s_1}} q(s_1,a) \right) + 0.2\left(-5 + \lambda \max_{a \in A_{s_2}} q(s_2,a) \right) \qquad (5.73) \\
&= 3 + 0.8\lambda \max_{a \in A_{s_1}} q(s_1,a) + 0.2\lambda \max_{a \in A_{s_2}} q(s_2,a), \qquad (5.74)
\end{aligned}$$

and since under $a_{1,2}$ the only possible transition is to s_2,

$$\begin{aligned}
q(s_1,a_{1,2}) &= p(s_2|s_1,a_{1,2})\left(r(s_1,a_{1,2},s_2) + \lambda \max_{a \in A_{s_2}} q(s_2,a) \right) \\
&= 5 + \lambda \max_{a \in A_{s_2}} q(s_2,a).
\end{aligned}$$

When $s = s_2$,

$$\begin{aligned}
q(s_2,a_{2,1}) &= p(s_2|s_2,a_{2,1})\left(r(s_2,a_{2,1},s_2) + \lambda \max_{a \in A_{s_2}} q(s_2,a) \right) \\
&= -5 + \lambda \max_{a \in A_{s_2}} q(s_2,a)
\end{aligned}$$

and

$$\begin{aligned} q(s_2, a_{2,2}) &= p(s_1|s_2, a_{2,2}) \left(r(s_2, a_{2,2}, s_1) + \lambda \max_{a \in A_{s_1}} q(s_1, a) \right) \\ &\quad + p(s_2|s_2, a_{2,2}) \left(r(s_2, a_{2,2}, s_2) + \lambda \max_{a \in A_{s_2}} q(s_2, a) \right) \\ &= 0.4 \left(20 + \lambda \max_{a \in A_{s_1}} q(s_1, a) \right) + 0.6 \left(-10 + \lambda \max_{a \in A_{s_2}} q(s_2, a) \right) \\ &= 2 + 0.4\lambda \max_{a \in A_{s_1}} q(s_1, a) + 0.6\lambda \max_{a \in A_{s_2}} q(s_2, a). \end{aligned}$$

Note that (5.73) is of the form (5.70) and (5.74) is of the form (5.69). The former would be the basis of a simulation model that generated $5 + \lambda \max_{a \in A_{s_1}} q(s_1, a)$ with probability 0.8 and $-5 + \lambda \max_{a \in A_{s_2}} q(s_2, a)$ with probability 0.2 corresponding to probabilities of transitions under action $a_{1,1}$ in s_1.

From Theorem 5.10 below, $q^*_\lambda(s, a)$ is the unique solution of these equations. It is left as an exercise to solve these equations and show that (5.65) holds.

5.4.4 Existence of solutions to the Bellman equation for state-action value functions*

Establishing the existence of solutions to either (5.66) or (5.69) follows the same contraction mapping approach as when the Bellman equation was expressed in terms of value functions.

Let V_q denote the set of real valued functions on $A_{s_1} \times A_{s_2} \times \ldots \times A_{s_M}$, or equivalently, indexed by the state s and action a. For an element $\mathbf{q} \in V_q$, define its norm by

$$\|\mathbf{q}\| = \max_{s \in S, a \in A_s} |q(s, a)|.$$

Note $\max_{s \in S, a \in A_s} |q(s, a)| = \max_{s \in S} \max_{a \in A_s} |q(s, a)|$.

Analogous to the operator $L : V \to V$, define the (s, a)-th component of $F : V_q \to V_q$ by

$$F\mathbf{q}(s, a) := r(s, a) + \lambda \sum_{j \in S} p(j|s, a) \max_{a' \in A_s} q(j, a'). \tag{5.75}$$

The following result establishes that the operator F is a contraction mapping on V_q so that as a result of the Banach fixed point theorem, there exists a unique solution to $F\mathbf{q} = \mathbf{q}$ in V_q.

Proposition 5.2. Suppose $0 \leq \lambda < 1$. Then the operator F defined in (5.75) is a contraction mapping on V_q with modulus λ.

Proof. Let $\mathbf{q}_1$ and $\mathbf{q}_2$ be elements of V_q. Then[16]

$$\begin{aligned}
\|F\mathbf{q}_1 - F\mathbf{q}_2\| &= \max_{s\in S, a\in A_s} \left| r(s,a) + \lambda \sum_{j\in S} p(j|s,a) \max_{a'\in A_j} q_1(j,a') - r(s,a) - \lambda \sum_{j\in S} p(j|s,a) \max_{a'\in A_j} q_2(j,a') \right| \\
&= \lambda \max_{s\in S, a\in A_s} \left| \sum_{j\in S} p(j|s,a) \max_{a'\in A_j} q_1(j,a') - \sum_{j\in S} p(j|s,a) \max_{a'\in A_j} q_2(j,a') \right| \\
&\leq \lambda \max_{s\in S, a\in A_s} \sum_{j\in S} p(j|s,a) \left| \max_{a'\in A_j} q_1(j,a') - \max_{a'\in A_j} q_2(j,a') \right| \\
&\leq \lambda \max_{s\in S, a\in A_s} \sum_{j\in S} p(j|s,a) \max_{a'\in A_j} |q_1(j,a') - q_2(j,a')| \\
&\leq \lambda \max_{s\in S, a\in A_s} \sum_{j\in S} p(j|s,a) \max_{j\in S} \max_{a'\in A_j} |q_1(j,a') - q_2(j,a')| \\
&= \lambda \|\mathbf{q}_1 - \mathbf{q}_2\|.
\end{aligned}$$

□

As a consequence of the above Proposition, applying the Banach fixed point theorem (Theorem 5.5) yields the following important result.

Theorem 5.10. Suppose $0 \leq \lambda < 1$.

1. Then the equation $F\mathbf{q} = \mathbf{q}$ has a unique solution $\mathbf{q}^* \in V_q$.
2. For any $\mathbf{q}^0 \in V_q$, the sequence $(\mathbf{q}^n)$, $n = 0, 1, \ldots$ defined by $\mathbf{q}^{n+1} = F\mathbf{q}^n = F^{n+1}\mathbf{q}^0$ converges to $\mathbf{q}^*$.

5.5 Spans, bounds and other technical concepts

This long and rather technical section serves as a bridge to the analysis of computational algorithms. The key results appear as Theorems 5.12, 5.13, and 5.14.

A formal statement of an iterative algorithm for solving an infinite horizon Markov decision process requires:

1. a stopping criterion,
2. an estimate of the optimal value function at termination, and
3. an estimate of the optimal policy at termination.

[16] The proof uses the inequality $\max_{a\in A} |f(a) - g(a)| \geq |\max_{a\in A} f(a) - \max_{a\in A} g(a)|$ for real-valued functions $f(a)$ and $g(a)$ on a set A. A proof of this result is left as an exercise.

Terminating such an algorithm requires determining when the difference in value between successive iterates is sufficiently small. This requires a concept of distance between two vectors. Section 5.1.6 introduced the norm $||\cdot||$; two vectors $\mathbf{u}$ and $\mathbf{v}$ are close if $||\mathbf{u}-\mathbf{v}||$ is small. However, using an alternative construct, referred to as the *span* of a vector, may detect convergence sooner.

Moreover, the span:

1. provides bounds on the distance of the value of a stationary policy derived from the "current" greedy decision rule to the optimal value function,
2. is the basis for identifying sub-optimal actions, and
3. provides enhanced estimates of the optimal value function.

5.5.1 Spans and span contractions

Definition 5.12. The *span semi-norm* or simply the *span*, $\text{sp}(\mathbf{v})$, is defined as

$$\text{sp}(\mathbf{v}) := \max_{s\in S} v(s) - \min_{s\in S} v(s). \tag{5.76}$$

It is easy to show that the span is a semi-norm[17]. Moreover, for $\mathbf{v} \in V$:

1. $\text{sp}(\mathbf{v}+k\mathbf{e}) = \text{sp}(\mathbf{v})$ for any scalar k,
2. $\text{sp}(\mathbf{v}) = \text{sp}(-\mathbf{v})$, and
3. $\text{sp}(\mathbf{v}) \leq 2\|\mathbf{v}\|$.

The first property means that $\text{sp}(\mathbf{v}) = 0$, implies that $\mathbf{v}$ is a constant vector. On the other hand, $\|\mathbf{v}\| = 0$, implies that $\mathbf{v} = 0$. Hence the span and the sup-norm measure different properties of a vector. The norm measures how close it is to $\mathbf{0}$ while the span measures how close it is to a constant vector. Note that the third condition is satisfied with equality when, for example, $\mathbf{v} = (1, -1)$.

[17] A *semi-norm* $\sigma(\mathbf{v})$ on V is a real-valued function that satisfies the following properties:

1. $\sigma(\mathbf{v}) \geq 0$ for all $\mathbf{v} \in V$,
2. $\sigma(\mathbf{v}+\mathbf{u}) \leq \sigma(\mathbf{v}) + \sigma(\mathbf{u})$ for $\mathbf{v}$ and $\mathbf{u}$ in V,
3. $\sigma(k\mathbf{v}) = |k|\sigma(\mathbf{v})$ for scalar k and $\mathbf{v} \in V$.

This is a semi-norm (and not a norm) because $\sigma(\mathbf{v}) = 0$ need not imply $\mathbf{v} = \mathbf{0}$.

Why the span is useful?

This section will show that under mild conditions, transition probability matrices have contraction properties with respect to the span. They are based on the following more general observation, which is stated without proof[18].

Proposition 5.3. Let $\mathbf{Q}$ denote an $K \times M$ matrix where each row of $\mathbf{Q}$ is probability distribution on $\{1, \ldots, M\}$. Then for any M-dimensional vector $\mathbf{v}$,

$$\text{sp}(\mathbf{Q}\mathbf{v}) \leq \gamma \text{sp}(\mathbf{v}) \tag{5.77}$$

where

$$\gamma = 1 - \min_{1 \leq k \leq K,\, 1 \leq k' \leq K} \sum_{m=1}^{M} \min\{Q(m|k), Q(m|k')\}. \tag{5.78}$$

Moreover, there exists a $\mathbf{v}$ for which (5.77) holds with equality.

When $\gamma < 1$ this result implies that the elements of $\mathbf{Q}\mathbf{v}$ will be closer together than those of $\mathbf{v}$. In other words $\mathbf{Q}\mathbf{v}$ is closer than $\mathbf{v}$ to a constant vector.

The main application of Proposition 5.3 in this book is when $\mathbf{Q} = \mathbf{P}_d$ so that $K = M = |S|$ and $\mathbf{Q}$ is a transition probability matrix corresponding to a Markovian decision rule d.

Proposition 5.4. For any $d \in D^{\text{MR}}$, transition probability matrix $\mathbf{P}_d : S \to S$, and $\mathbf{v} \in V$,

$$\text{sp}(\mathbf{P}_d \mathbf{v}) \leq \gamma_d \text{sp}(\mathbf{v}), \tag{5.79}$$

where

$$\gamma_d = \quad 1 - \min_{(s,u) \in S \times S} \sum_{j \in S} \min\left\{P_d(j|s), P_d(j|u)\right\} \tag{5.80}$$

Moreover, there exists a $\mathbf{v} \in V$ for which (5.79) holds with equality.

The quantity γ_d is referred to as the *delta coefficient* of $\mathbf{P}_d$. It provides an upper bound on the second largest eigenvalue (*sub-radius*) of $\mathbf{P}_d$. Note that (5.80) is not intended for computation[19] but rather to provide insight into the meaning of γ_d. In

[18] A slightly less general version of this result is stated and proved in Hübner [1977] and restated as Proposition 6.6.1 in Puterman [1994].

[19] Alternatively

$$\gamma_d = \frac{1}{2} \max_{(s,u) \in S \times S} \sum_{j \in S} |P_d(j|s) - P_d(j|u)| = \max_{(s,u) \in S \times S} \sum_{j \in S} (P_d(j|s) - P_d(j|u))^+.$$

particular, when $\mathbf{P}_d = \mathbf{I}$ or when $\mathbf{P}_d$ has exactly one 1 in each row and column, direct computation using (5.80) establishes that $\gamma_d = 1$. On the other hand if the rows of $\mathbf{P}_d$ are equal, it is easy to see then $\gamma_d = 0$.

Note that the following simpler expression γ'_d provides an easier to compute upper bound on γ_d:

$$\gamma_d \leq \gamma'_d := 1 - \sum_{j \in S} \min_{s \in S} P_d(j|s). \tag{5.81}$$

From Lemma 5.1, it follows for any Markovian decision rule that

$$\|\mathbf{P}_d \mathbf{v}\| \leq ||\mathbf{P}_d||\, ||\mathbf{v}|| = \|\mathbf{v}\|,$$

so in contrast to the span, $\mathbf{P}_d$ is **not** a contraction with respect to the norm.

Example 5.7. This example computes γ_d for the Markov deterministic decision rule $d(s_1) = a_{1,1}$, $d(s_2) = a_{2,1}$ in the two-state model. Since

$$\mathbf{P}_d = \begin{bmatrix} 0.8 & 0.2 \\ 0 & 1 \end{bmatrix},$$

for any $\mathbf{v} = (v_1, v_2)$,

$$\mathbf{P}_d \mathbf{v} = \begin{bmatrix} 0.8v_1 + 0.2v_2 \\ v_2 \end{bmatrix}.$$

Hence

$$\text{sp}(\mathbf{P}_d \mathbf{v}) = |0.8v_1 - 0.8v_2| = 0.8|v_1 - v_2| = 0.8\, \text{sp}(\mathbf{v}).$$

Thus

$$\text{sp}(\mathbf{P}_d^n \mathbf{v}) = 0.8^n \text{sp}(\mathbf{v})$$

so that $\text{sp}(\mathbf{P}_d^n \mathbf{v})$ converges to $\mathbf{0}$ for any $\mathbf{v} \in V$.

Direct computation shows that $\gamma_d = \gamma'_d = 1 - (0 + 0.2) = 0.8$ in agreement with the above calculations. Note also that the eigenvalues of $\mathbf{P}_d$ equal 1 and 0.8 and that γ_d equals the second largest eigenvalue of $\mathbf{P}_d$[a].

It is left to the reader to compute the values of γ_d and γ'_d, and the corresponding eigenvalues for the three other Markovian deterministic decision rules.

[a]For any square matrix, the sum of the diagonal elements (the *trace*) is equal to the sum of its eigenvalues. Since 1 is an eigenvalue of any finite transition probability matrix, for a 2×2 transition probability matrix one can find the second eigenvalue by subtracting 1 from its trace.

The following theorem identifies broad classes of transition probability matrices for which $\gamma_d < 1$. It is one of the few results for discounted models that depend on the structure of $\mathbf{P}_d$. It is a consequence of the Perron-Frobenius Theorem, which appears as Theorem B.3 in the book Appendix.

Proposition 5.5. Let $d \in D^{\text{MD}}$. Suppose the Markov chain corresponding to $\mathbf{P}_d$ is either[a]

1. regular or
2. has a single irreducible set and a non-empty set of transient states.

Then $\gamma_d < 1$.

[a]See Appendix B for definitions of the concepts below.

Note that the matrices

$$\mathbf{P}_d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{or} \quad \mathbf{P} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

do not satisfy the hypotheses of Theorem 5.5. The one on the left has two closed classes and the one on the right is periodic so it is not regular. On the other hand the matrix in Example 5.7 satisfies the second condition in this theorem.

Span contraction mappings

Establishing that L is a span contraction with modulus at most λ requires the following application of Proposition 5.3 with a different choice for $\mathbf{Q}$ defined in the following proposition.

Proposition 5.6. Let $\mathscr{P}$ represent a $\sum_{s\in S} |A_s| \times |S|$ matrix[a] where each row consists of the vector of probabilities $p(\cdot|s,a)$ for a different (s,a)-pair where $s \in S$ and $a \in A_s$. Then for any $\mathbf{v} \in V$

$$\text{sp}(\mathscr{P}\mathbf{v}) \leq \gamma^* \text{sp}(\mathbf{v}), \tag{5.82}$$

where

$$\gamma^* = 1 - \min_{s\in S, a\in A_s, s'\in S, a'\in A_{s'}} \sum_{j\in S} \min\{p(j|s,a), p(j|s',a')\} \leq 1. \tag{5.83}$$

Moreover, there exists a $\mathbf{v} \in V$ for which (5.82) holds with equality.

[a]The order of the rows is material for what follows.

The following theorem is the main result of this section. Its proof is quite subtle but uses some of the same arguments that were the basis for establishing that L was a sup-norm contraction.

Theorem 5.11. For any $\mathbf{v}$ and $\mathbf{u}$ in V, and $0 \leq \lambda < 1$, there exists $\gamma^* \leq 1$ such that

$$\mathrm{sp}(L\mathbf{v} - L\mathbf{u}) \leq \lambda\gamma^* \mathrm{sp}(\mathbf{v} - \mathbf{u}). \tag{5.84}$$

Proof. Choose $\mathbf{v}$ and $\mathbf{u}$ in V and let Markovian deterministic decision rule $d_{\mathbf{v}}$ and $d_{\mathbf{u}}$ be greedy with respect to $\mathbf{v}$ and $\mathbf{u}$ respectively. Then

$$L\mathbf{v} - L\mathbf{u} \leq L_{d_{\mathbf{v}}}\mathbf{v} - L_{d_{\mathbf{v}}}\mathbf{u} = \lambda \mathbf{P}_{d_{\mathbf{v}}}(\mathbf{v} - \mathbf{u})$$

and

$$L\mathbf{v} - L\mathbf{u} \geq L_{d_{\mathbf{u}}}\mathbf{v} - L_{d_{\mathbf{u}}}\mathbf{u} = \lambda \mathbf{P}_{d_{\mathbf{u}}}(\mathbf{v} - \mathbf{u}).$$

Since $P_{d_{\mathbf{v}}}$ and $P_{d_{\mathbf{u}}}$ consist[20] of rows of $\mathscr{P}$, it follows that

$$\max_{s\in S}\big(L\mathbf{v}(s) - L\mathbf{u}(s)\big) \leq \lambda \max_{s\in S} \mathbf{P}_{d_{\mathbf{v}}}(\mathbf{v} - \mathbf{u}) \leq \lambda \max_{s\in S} \mathscr{P}(\mathbf{v} - \mathbf{u})$$

and

$$\min_{s\in S}\big(L\mathbf{v}(s) - L\mathbf{u}(s)\big) \geq \lambda \min_{s\in S} \mathbf{P}_{d_{\mathbf{u}}}(\mathbf{v} - \mathbf{u}) \geq \lambda \min_{s\in S} \mathscr{P}(\mathbf{v} - \mathbf{u}).$$

Combining these two inequalities and applying Proposition 5.6 gives the desired result that

$$\mathrm{sp}(L\mathbf{v} - L\mathbf{u}) \leq \lambda\, \mathrm{sp}(\mathscr{P}(\mathbf{v} - \mathbf{u})) \leq \lambda\gamma^*\, \mathrm{sp}(\mathbf{v} - \mathbf{u}).$$

□

Note that in most settings, γ^* cannot be easily determined *a priori*. Moreover, there is no need to do so. When either of the conditions in Theorem 5.5 holds and the sets of states and action are finite, $\gamma^* < 1$. Hence, in this case *the contraction modulus with respect to the span semi-norm is strictly less than that of the sup-norm* that is, $\lambda\gamma^* < \lambda$. This suggests that algorithms of the form $\mathbf{v} \leftarrow L\mathbf{v}$ will terminate faster when using the span as a stopping criterion.

But when $\gamma^* = 1$, as long as $\lambda < 1$, L will remain a span contraction but with modulus possibly equal to λ. Note the subtle point that algorithms of the form $\mathbf{v} \leftarrow L\mathbf{v}$ might indeed converge faster with respect to the span because the series of greedy decision rules chosen might all have $\gamma_d < 1$ even when $\gamma^* = 1$.

The following example illustrates these calculations.

[20] Since $\mathbf{P}_{d_{\mathbf{v}}}$ need not equal $\mathbf{P}_{d_{\mathbf{u}}}$ the use of $\mathscr{P}$ is required to ensure that the maximum and minimum are with respect to the same quantity.

Example 5.8. In the two-state model

$$\mathscr{P} = \begin{bmatrix} 0.8 & 0.2 \\ 0 & 1 \\ 0 & 1 \\ 0.6 & 0.4 \end{bmatrix} \tag{5.85}$$

where the first two rows correspond to $(s_1, a_{1,1})$ and $(s_1, a_{1,2})$ and the last two rows to $(s_2, a_{2,1})$ and $(s_2, a_{2,2})$. Direct calculation shows that $\gamma^* = 0.8$. Hence the contraction modulus with respect to the span is 0.8λ, while that with respect to the norm is λ.

Restricting the set of possible decision rules to a single decision rule d leads to the following corollary, which can easily be proved directly.

Corollary 5.4. For any $d \in D^{\text{MD}}$, $\mathbf{v}$ and $\mathbf{u}$ in V, and $0 \leq \lambda < 1$,

$$\text{sp}(L_d\mathbf{v} - L_d\mathbf{u}) \leq \lambda\gamma_d \text{sp}(\mathbf{v} - \mathbf{u}), \tag{5.86}$$

where γ_d is defined in (5.80).

5.5.2 Properties of $L\mathbf{v} - \mathbf{v}$

The quantity $L\mathbf{v} - \mathbf{v}$ plays an important role in deriving bounds, eliminating actions and proving convergence of several iterative algorithms. Since it arises so frequently, the operator B is defined to represent it.

Definition 5.13. The *optimal one-step improvement operator* is the operator $B : V \to V$ defined by

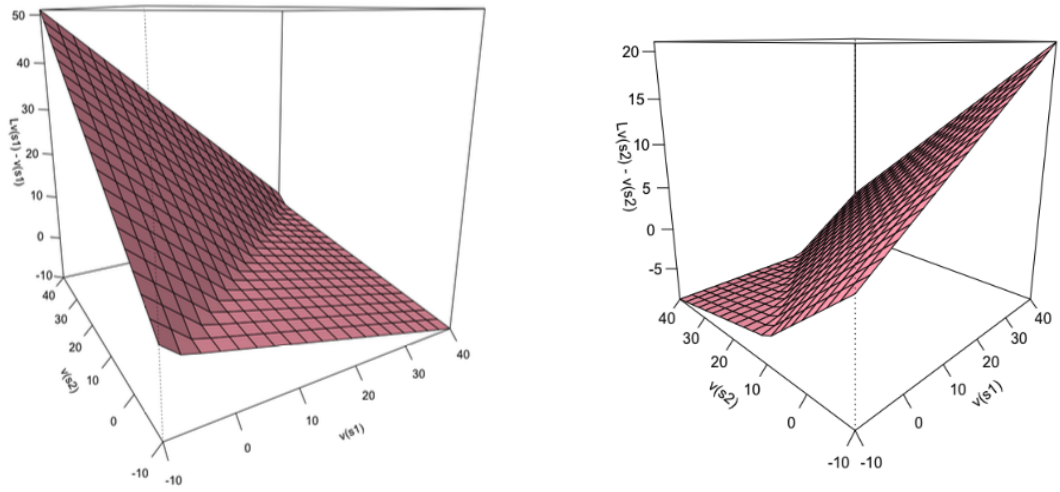
$$B\mathbf{v} := L\mathbf{v} - \mathbf{v}. \tag{5.87}$$

Note that the Bellman equation $L\mathbf{v} = \mathbf{v}$ can now be expressed as

$$B\mathbf{v} = \mathbf{0} \tag{5.88}$$

so that solving the Bellman equation may be regarded as finding a zero of B. This observation will be important when analyzing policy iteration below. For completeness, define the operator B_d on V by $B_d := L_d\mathbf{v} - \mathbf{v}$. Using this notation, $B\mathbf{v}$ can be rewritten as $B\mathbf{v} = \text{c-max}_{d \in D^{\text{MD}}} B_d\mathbf{v}$.

Figure 5.3 shows *each* component of $B\mathbf{v}$ as a function of the components of $\mathbf{v}$ for the two-state model. Because there are only two states, the components of $\mathbf{v}$ can be



(a) $B\mathbf{v}(s_1)$ as a function of $v(s_1)$ and $v(s_2)$. (b) $B\mathbf{v}(s_2)$ as a function of $v(s_1)$ and $v(s_2)$.

Figure 5.3: Each components of $B\mathbf{v}$ as a function of the components of $\mathbf{v}$ for the two-state example.

shown on the horizontal axes and the value of **each** component of $B\mathbf{v}$ on the vertical axis in a separate figure.

Observe that in each case the surfaces represent the maximum of two hyperplanes, each corresponding to a different action. Hence each component is piecewise linear and convex. Moreover $B\mathbf{v}(s_1)$ is non-decreasing as a function of $v(s_1)$ for each fixed value of $v(s_2)$ and $B\mathbf{v}(s_2)$ is non-increasing as a function of $v(s_2)$ for each fixed value of $v(s_1)$.

To get another perspective on the form of $B\mathbf{v}$, Figure 5.4 provides a one-dimensional plot of $B\mathbf{v}(s_1)$ as a function of $v(s_1)$ when $v(s_2)$ equals its optimal value 27.941. Observe that it is the maximum of two lines, each corresponding to one of the deterministic decision rules and that it is convex non-increasing in $v(s_1)$ in agreement with Figure 5.3a. The value of each deterministic decision rule can be found by solving $B_d\mathbf{v}(s_1) = 0$ for $d = d_1$ and d_2, and the optimal value by solving $B\mathbf{v}(s_1) = 0$. This figure also shows that the decision rule d_1 (dotted line) corresponds to the optimal policy.

Convexity of $B\mathbf{v}$

The following lemma, which generalizes the gradient inequality for convex functions[21], establishes convexity-like properties of B and L. Expression (5.89) is often called the

[21] If $f(x)$ is a convex and differentiable function mapping $\mathbb{R}^1$ into $\mathbb{R}^1$, then for any x and y in $\mathbb{R}^1$,

$$f(y) \geq f(x) + f'(x)(y - x)$$

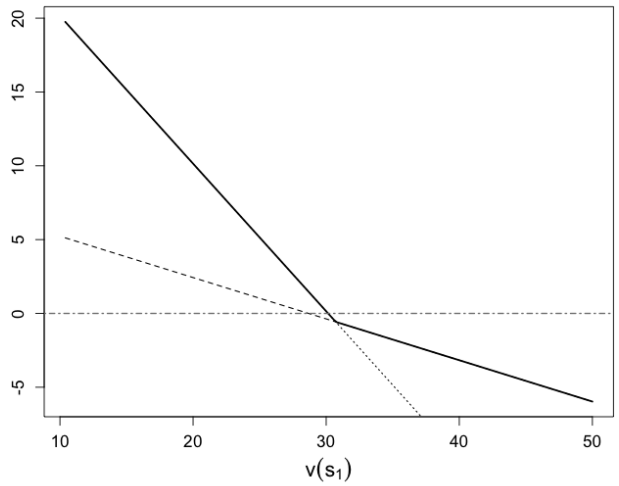


Figure 5.4: Plot showing $B\mathbf{v}(s_1)$ (solid line), $B_{d_1}\mathbf{v}(s_1)$ (dotted line corresponding to $d_1 = (a_{1,2}, a_{2,2})$) and $B_{d_2}\mathbf{v}(s_1)$ (dashed line corresponding to $d_2 = (a_{1,1}, a_{2,2})$) as a function of $v(s_1)$ for the two-state model when $v(s_2)$ is set to its optimal value 27.941. The optimal value $v(s_1) = 30.147$ is obtained when $Bv(s_1) = 0$.

support inequality.

Lemma 5.5. For any $\mathbf{u}$ and $\mathbf{v}$ in V, and $d_{\mathbf{v}}$ that satisfies $\mathbf{r}_{d_{\mathbf{v}}} + \lambda \mathbf{P}_{d_{\mathbf{v}}}\mathbf{v} = L\mathbf{v}$,

$$B\mathbf{u} \geq B\mathbf{v} + (\lambda \mathbf{P}_{d_{\mathbf{v}}} - \mathbf{I})(\mathbf{u} - \mathbf{v}), \tag{5.89}$$

or equivalently,

$$L\mathbf{u} - \mathbf{u} \geq L\mathbf{v} - \mathbf{v} + (\lambda \mathbf{P}_{d_{\mathbf{v}}} - \mathbf{I})(\mathbf{u} - \mathbf{v}). \tag{5.90}$$

Proof. By the definition of $d_{\mathbf{v}}$,

$$L\mathbf{u} \geq \mathbf{r}_{d_{\mathbf{v}}} + \lambda \mathbf{P}_{d_{\mathbf{v}}}\mathbf{u} = \mathbf{r}_{d_{\mathbf{v}}} + \lambda \mathbf{P}_{d_{\mathbf{v}}}\mathbf{v} + \lambda \mathbf{P}_{d_{\mathbf{v}}}(\mathbf{u} - \mathbf{v}) = L\mathbf{v} + \lambda \mathbf{P}_{d_{\mathbf{v}}}(\mathbf{u} - \mathbf{v})$$

subtracting $\mathbf{u} - \mathbf{v}$ from both sides and rearranging terms gives the result. □

5.5.3 Bounds

Bounds will be used to determine how accurately $\mathbf{v}^n$ (the n-th iterate of an algorithm) approximates $\mathbf{v}^*_\lambda$ when an algorithm terminates, although they apply in greater generality. They will also provide the basis for action elimination procedures in Section 5.5.5.

Some additional notation and a definition simplify exposition. For $\mathbf{v} \in V$, define

the scalars[22]:

$$\underline{\mathbf{v}} := \min_{s \in S} v(s) \quad \text{and} \quad \overline{\mathbf{v}} := \max_{s \in S} v(s). \tag{5.91}$$

Definition 5.14. For any $\mathbf{v} \in V$, $d_{\mathbf{v}} \in D^{\text{MR}}$ is $\mathbf{v}$-*greedy* if

$$\mathbf{r}_{d_{\mathbf{v}}} + \lambda \mathbf{P}_{d_{\mathbf{v}}} \mathbf{v} = L\mathbf{v}. \tag{5.92}$$

Note that $d_{\mathbf{v}} \in D^{\text{MD}}$ is $\mathbf{v}$-greedy if

$$d_{\mathbf{v}} \in \arg\operatorname*{c\text{-}max}_{d \in D^{\text{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}.$$

However, the definition also holds for randomized decision rules that obtain the componentwise maximums.

The following theorem provides two sets of upper and lower bounds on the optimal value function and the value function of a $\mathbf{v}$-greedy policy. It is the main result in this section.

Theorem 5.12. For any $\mathbf{v} \in V$, and $\mathbf{v}$-greedy $d_{\mathbf{v}} \in D^{\text{MR}}$

$$\begin{aligned} \mathbf{v} + (1-\lambda)^{-1}(\overline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \geq L\mathbf{v} + \lambda(1-\lambda)^{-1}(\overline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v}^*_\lambda \geq \mathbf{v}^{d^\infty_{\mathbf{v}}}_\lambda \\ \geq L\mathbf{v} + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v} + (1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \end{aligned} \tag{5.93}$$

Proof. A derivation of the lower bounds in (5.93) follows. The upper bounds can be obtained by reversing $\mathbf{v}$ and $\mathbf{v}^*_\lambda$ in the following.

Applying Lemma 5.5 with $\mathbf{u} = \mathbf{v}^*_\lambda$ and noting that $L\mathbf{v}^*_\lambda = \mathbf{v}^*_\lambda$ gives

$$\mathbf{0} \geq L\mathbf{v} - \mathbf{v} + (\lambda \mathbf{P}_{d_{\mathbf{v}}} - \mathbf{I})(\mathbf{v}^*_\lambda - \mathbf{v}).$$

From part 1 of Lemma 5.3, for $\mathbf{w} \in V$, $(\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{v}}})^{-1}\mathbf{w} \leq \mathbf{0}$ when $\mathbf{w} \leq \mathbf{0}$ so that left-multiplying both sides of the above expression by $(\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{v}}})^{-1}$ and rearranging terms yields

$$\begin{aligned} \mathbf{v}^*_\lambda &\geq \mathbf{v} + (\mathbf{I} - \lambda \mathbf{P}_{d_v})^{-1}(L\mathbf{v} - \mathbf{v}) && (5.94)\\ &= \mathbf{v} + (\mathbf{I} + \lambda \mathbf{P}_{d_{\mathbf{v}}} + \lambda^2 \mathbf{P}^2_{d_{\mathbf{v}}} + \ldots)(L\mathbf{v} - \mathbf{v}) && (5.95)\\ &= \mathbf{v} + (L\mathbf{v} - \mathbf{v}) + \lambda \mathbf{P}_{d_{\mathbf{v}}}(\mathbf{I} + \lambda \mathbf{P}_{d_{\mathbf{v}}} + \lambda^2 \mathbf{P}^2_{d_{\mathbf{v}}} + \ldots)(L\mathbf{v} - \mathbf{v}) && (5.96)\\ &\geq \mathbf{v} + (L\mathbf{v} - \mathbf{v}) + \lambda \mathbf{P}_{d_{\mathbf{v}}}(\mathbf{I} + \lambda \mathbf{P}_{d_{\mathbf{v}}} + \lambda^2 \mathbf{P}^2_{d_{\mathbf{v}}} + \ldots)(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} && (5.97)\\ &= \mathbf{v} + (L\mathbf{v} - \mathbf{v}) + \lambda \mathbf{P}_{d_{\mathbf{v}}}(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} && (5.98) \end{aligned}$$

[22] The notation here is a bit awkward. It is best understood by viewing $\underline{\mathbf{v}}$ and $\overline{\mathbf{v}}$ as real-valued functions applied to the vector $\mathbf{v}$.

$$= \mathbf{v} + (L\mathbf{v} - \mathbf{v}) + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \tag{5.99}$$
$$\geq \mathbf{v} + (1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e}. \tag{5.100}$$

Equality (5.95) is an immediate consequence of Theorem 5.3 and (5.96) follows by rewriting (5.95). Inequality (5.97) follows by noting $L\mathbf{v} - \mathbf{v}$ is lower bounded by $(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e}$. Equation (5.98) follows from the third part of Lemma 5.3. Equation (5.99) follows from the fact the rows of $\mathbf{P}_d^n$ sum to 1, i.e., $\mathbf{P}_d^n\mathbf{e} = \mathbf{e}$, for all $n \geq 0$. Noting that the terms involving $\mathbf{v}$ cancel in (5.99) yields the inner lower bound in (5.93). Equation (5.100) follows by replacing $L\mathbf{v} - \mathbf{v}$ in (5.99) by $(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e}$.

Rewriting the right hand side of (5.94) and using the assumption that $d_\mathbf{v}$ is $\mathbf{v}$-greedy gives

$$\mathbf{v}+(\mathbf{I}-\lambda\mathbf{P}_{d_\mathbf{v}})^{-1}(L\mathbf{v}-\mathbf{v}) = \mathbf{v}+(\mathbf{I}-\lambda\mathbf{P}_{d_\mathbf{v}})^{-1}(\mathbf{r}_{d_\mathbf{v}}+(\lambda\mathbf{P}_{d_\mathbf{v}}-\mathbf{I})\mathbf{v}) = (\mathbf{I}-\lambda\mathbf{P}_{d_\mathbf{v}})^{-1}\mathbf{r}_{d_\mathbf{v}} = \mathbf{v}_\lambda^{d_\mathbf{v}^\infty},$$

implying

$$\mathbf{v}_\lambda^{d_v^\infty} \geq L\mathbf{v} + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e}.$$

□

Some comments about the bounds in the above theorem follow:

1. Several generalizations of these bounds are possible by splitting (5.96) at higher order terms, but those above are most relevant to what follows.

2. The lower bound based on $\mathbf{v}_\lambda^{d_\mathbf{v}^\infty}$ is also an obvious consequence of the definition of $\mathbf{v}_\lambda^*$. It is not intended for direct evaluation but instead enables you to understand how close $d_\mathbf{v}$ is to optimal.

3. Computing the tighter inner bound requires no additional computational effort since one must evaluate $L\mathbf{v}$ to obtain the looser outer bound.

The following calculations illustrate computation of these bounds in the two-state example.

Example 5.9. From Example 5.5,

$$L\mathbf{v} = \begin{bmatrix} \max\{3 + \lambda(0.8v(s_1) + 0.2v(s_2)), 5 + \lambda v(s_2)\} \\ \max\{-5 + \lambda v(s_2), 2 + \lambda(0.4v(s_1) + 0.6v(s_2))\} \end{bmatrix}.$$

Setting $\lambda = 0.9$ and choosing $\mathbf{v} = (5, -5)$ gives

$$L\mathbf{v} = \begin{bmatrix} \max\{5.7, 0.5\} \\ \max\{-9.5, 1.1\} \end{bmatrix} = \begin{bmatrix} 5.7 \\ 1.1 \end{bmatrix}.$$

Hence $L\mathbf{v} - \mathbf{v} = (0.7, 6.1)$ and $\underline{L\mathbf{v} - \mathbf{v}} = 0.7$ and $\overline{L\mathbf{v} - \mathbf{v}} = 6.1$. Moreover $d_\mathbf{v} = (a_{1,1}, a_{2,2})$ is $\mathbf{v}$-greedy.

The bounds in (5.93) become

$$\begin{bmatrix}66\\56\end{bmatrix} \geq \begin{bmatrix}60.6\\56\end{bmatrix} \geq \mathbf{v}_\lambda^* \geq \begin{bmatrix}27.188\\25.625\end{bmatrix} \geq \begin{bmatrix}12\\7.4\end{bmatrix} \geq \begin{bmatrix}12\\2\end{bmatrix}$$

Since $\mathbf{v}_\lambda^* = (30.147, 27.941)$ neither bound based on $L\mathbf{v}-\mathbf{v}$ is close to the optimal value but the second bound is tighter. Note also that the average of the upper and lower bounds provides an enhanced estimate of $\mathbf{v}_\lambda^*$. It is left as an exercise to evaluate these bounds for other choices of $\mathbf{v}$.

5.5.4 Stopping criteria

Bounds provide a guarantee on the quality of an approximation when terminating an algorithm. The following result will apply to value iteration and any other algorithm that evaluates $L\mathbf{v}$.

Theorem 5.13. For any $\mathbf{v} \in V$ and $\epsilon > 0$, if

$$\mathrm{sp}(L\mathbf{v} - \mathbf{v}) < \frac{1-\lambda}{\lambda}\epsilon \tag{5.101}$$

then

$$\|L\mathbf{v} + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v}-\mathbf{v}})\mathbf{e} - \mathbf{v}_\lambda^*\| < \epsilon \tag{5.102}$$

and for any $\mathbf{v}$-greedy $d_\mathbf{v} \in D^{\mathrm{MR}}$

$$\|\mathbf{v}_\lambda^{d_\mathbf{v}^\infty} - \mathbf{v}_\lambda^*\| < \epsilon. \tag{5.103}$$

Proof. Note that for any vectors satisfying $\mathbf{x} \geq \mathbf{y} \geq \mathbf{z} \geq \mathbf{w}$, it follows that $\mathbf{x}-\mathbf{w} \geq \mathbf{y}-\mathbf{z} \geq \mathbf{0}$ and $\mathbf{x}-\mathbf{w} \geq \mathbf{y}-\mathbf{w} \geq \mathbf{0}$. Consequently $\|\mathbf{x}-\mathbf{w}\| \geq \|\mathbf{y}-\mathbf{z}\|$ and $\|\mathbf{x}-\mathbf{w}\| \geq \|\mathbf{y}-\mathbf{w}\|$.

Equations (5.102) and (5.103) follow from (5.93) by setting $\mathbf{x} = L\mathbf{v} + \lambda(1-\lambda)^{-1}(\overline{L\mathbf{v}-\mathbf{v}})\mathbf{e}$, $\mathbf{y} = \mathbf{v}_\lambda^*$, $\mathbf{z} = \mathbf{v}_\lambda^{d_\mathbf{v}^\infty}$ and $\mathbf{w} = L\mathbf{v} + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v}-\mathbf{v}})\mathbf{e}$, applying the above inequalities involving $\mathbf{x}, \mathbf{y}, \mathbf{z}$ and $\mathbf{w}$ and noting that $\mathrm{sp}(L\mathbf{v}-\mathbf{v}) = \overline{L\mathbf{v}-\mathbf{v}} - (\underline{L\mathbf{v}-\mathbf{v}})$ and $\|\mathbf{e}\| = 1$. □

Since most iterative algorithms will not generate a direct estimate of $\mathbf{v}_\lambda^{d_\mathbf{v}^\infty}$, the easily computable estimate referred to as the *lower bound extrapolation*,

$$L\mathbf{v} + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v}-\mathbf{v}})\mathbf{e},$$

well approximates $\mathbf{v}_\lambda^*$ when (5.101) holds for some small $\epsilon > 0$. However even without using this estimate, it still follows that $d_\mathbf{v}^\infty$ is ϵ-optimal although its value may not be determined without further calculations.

5.5.5 Action elimination

This section begins with the following obvious result, which provides the basis for eliminating non-optimal actions.

Proposition 5.7. Suppose in state $s \in S$,

$$a' \notin \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_\lambda^*(j) \right\}$$

or equivalently

$$r(s,a') + \lambda \sum_{j \in S} p(j|s,a') v_\lambda^*(j) < \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_\lambda^*(j) \right\}. \quad (5.104)$$

Then any stationary policy that selects a' in state s with positive probability cannot be optimal.

Since $\mathbf{v}_\lambda^*$ is unknown, (5.104) cannot be used to identify non-optimal actions. However judiciously replacing it by upper and lower bounds leads to the following practical result.

Theorem 5.14. Suppose $\mathbf{v}^L \leq \mathbf{v}_\lambda^* \leq \mathbf{v}^U$. If

$$r(s,a') + \lambda \sum_{j \in S} p(j|s,a') v^U(j) < v^L(s), \quad (5.105)$$

any stationary policy that selects a' in state s with positive probability cannot be optimal.

Proof. Suppose (5.105) holds. Then

$$r(s,a') + \lambda \sum_{j \in S} p(j|s,a') v_\lambda^*(j) \leq r(s,a') + \lambda \sum_{j \in S} p(j|s,a') v^U(j)$$

$$< v^L(s) \leq v_\lambda^*(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_\lambda^*(j) \right\}.$$

Hence the result follows from Proposition 5.7 □

The previous section provided explicit bounds that can be used to eliminate non-optimal actions. Substituting them into (5.105) gives the main result of this section.

Theorem 5.15. Suppose

$$\frac{\lambda}{(1-\lambda)}\text{sp}(L\mathbf{v}-\mathbf{v}) < L\mathbf{v}(s) - \left(r(s,a') + \lambda\sum_{j\in S} p(j|s,a')v(j)\right) \tag{5.106}$$

for some $\mathbf{v} \in V$. Then any stationary policy that selects a' in state s with positive probability cannot be optimal.

Proof. Substitute the outer upper bound and inner lower bound from (5.93) into (5.105) gives

$$r(s,a') + \lambda\sum_{j\in S} p(j|s,a')\Big(v(j)+(1-\lambda)^{-1}(\overline{L\mathbf{v}-\mathbf{v}})\Big) < L\mathbf{v}(s) + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v}-\mathbf{v}}).$$

Rearranging terms and applying Proposition 5.7 gives the result. □

Some comments about this result follow:

1. Note that the choice of bounds in (5.106) is guided by avoiding additional computation to evaluate
$$r(s,a') + \lambda\sum_{j\in S} p(j|s,a')L\mathbf{v}(j),$$
which would have been necessary if instead the tighter upper bound in (5.93) had been used.

2. By organizing calculations efficiently, the right hand side of (5.106) can be evaluated with minimal extra effort.

3. The next section will show how to integrate this result into value iteration.

4. The literature also proposes action elimination methods that identify actions for temporary elimination, that is, they will not be part of a $\mathbf{v}^n$-greedy decision rule, where $\mathbf{v}^n$ is the n-th iterate of an algorithm but could be in a $\mathbf{v}^m$-greedy decision rule for some $m \geq n$.

This section concludes with an example.

Example 5.10. Continuing with the calculations in Example 5.9 let

$$q(s,a') := L\mathbf{v}(s) - \left(r(s,a') + \lambda\sum_{j\in S} p(j|s,a')v(j)\right)$$

denote the quantity on the right hand side of (5.106). Start with $\mathbf{v}^0 = (5,-5)$

and compute $\mathbf{v}^1 = L\mathbf{v}^0$ and $\mathbf{v}^2 = L\mathbf{v}^1$ and in each case evaluate the expressions on either side of (5.106). Results are given in the following table.

n	$\mathbf{v}^n$	$\frac{\lambda}{1-\lambda}\text{sp}(L\mathbf{v}^n - \mathbf{v}^n)$	$q(s_1, a_{1,1})$	$q(s_1, a_{1,2})$	$q(s_2, a_{2,1})$	$q(s_2, a_{2,2})$
0	$(5, -5)$	48.6	0	5.2	10.6	0
1	$(5.7, 1.1)$	17.50	0	1.31	8.66	0
2	$(7.30, 4.65)$	**5.51**	0.09	0	**7.96**	0

Observe that when $\mathbf{v} = \mathbf{v}^2$, the inequality in (5.106) holds in state s_2 with $a' = a_{2,1}$ as denoted by bold text. Hence this action cannot be optimal in state 2 and at subsequent iterations does not need to be evaluated.

Continuing the above calculations shows that when $\mathbf{v} = \mathbf{v}^5 = (13.24, 11.06)$, action $a_{1,1}$ is eliminated in s_1. Hence six iterations of $\mathbf{v}^{n+1} = L\mathbf{v}^n$, establish that the stationary policy d^∞ with $d = (a_{1,2}, a_{2,2})$ is optimal. This is very surprising because $\mathbf{v}^5$ is quite far from the optimal value $\mathbf{v}^*_\lambda = (30.147, 27.941)$. Note further that that value obtained by the lower bound extrapolation in (5.102) equals $(30.08, 27.87)$, which is a close approximation to $\mathbf{v}^*_\lambda$.

This example illustrates how to incorporate action elimination in an iterative procedure. It shows that the optimal policy has been identified when all but one action in each state has been eliminated. While this may be the case in certain situations, in our experience, action elimination methods are not always so rewarding.

5.6 Value iteration

Value iteration is the most widely used and easiest to implement method for solving discounted models. It generalizes the standard successive approximations approach for solving fixed point equations of the form $f(x) = x$. It is frequently referred to as *backwards induction*, *successive approximation* or even *dynamic programming*.

5.6.1 A value iteration algorithm

Figure 5.5 provides a schematic representation of a value iteration algorithm. Observe its simplicity: it repeats $\mathbf{v}' \leftarrow L\mathbf{v}$ until a stopping criterion (described below) is satisfied.

The following algorithmic statement formalizes the above flow chart. It combines all steps in the loop and specifies how to choose a policy and estimate its value at termination. The stopping criterion will be discussed subsequently.

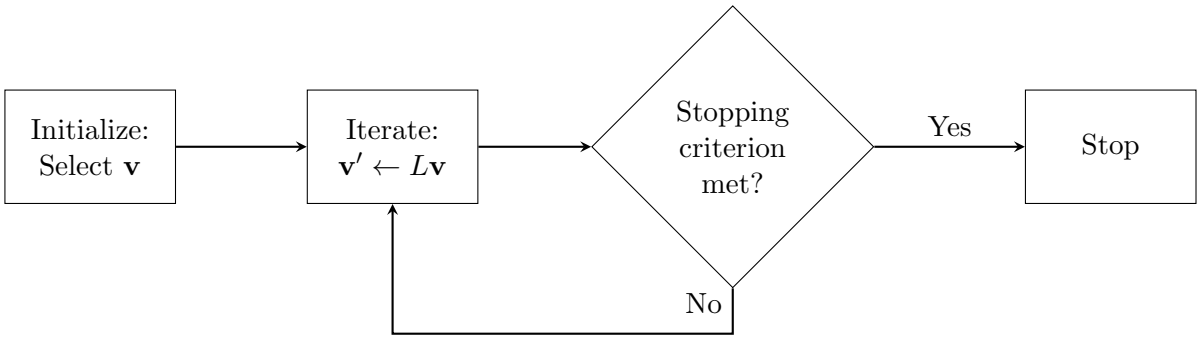


Figure 5.5: Flow diagram for value iteration

Algorithm 5.2. Value iteration: vector form

1. **Initialize:** Specify $\mathbf{v} \in V$ and a stopping criterion that involves an $\epsilon > 0$.
2. **Iterate:** Do until stopping criterion met:
$$\mathbf{v}' \leftarrow L\mathbf{v}.$$
3. **Choose an ϵ-optimal policy:** Select
$$d_\epsilon \in \arg\operatorname*{c\text{-}max}_{d \in D^{\mathrm{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}.$$
4. **Approximate the optimal value function:** Set
$$\mathbf{v}_\lambda^\epsilon \leftarrow \mathbf{v}' + \frac{\lambda}{1-\lambda}(\underline{\mathbf{v}' - \mathbf{v}})\mathbf{e}.$$

In practice, value iteration is applied component-wise as follows. Iterates in the algorithm below are represented by superscripts on the value function to remain consistent with this book's use of subscripts to denote decision epochs or specific states.

Algorithm 5.3. Value iteration: component form

1. **Initialize:**
   (a) Specify $v'(s)$ for all $s \in S$.
   (b) Specify $\epsilon > 0$ and $\sigma > (1-\lambda)\epsilon/\lambda$.

2. **Iterate:** While
$$\sigma \geq \frac{1-\lambda}{\lambda}\epsilon : \tag{5.107}$$
   (a) $v(s) \leftarrow v'(s)$ for all $s \in S$.
   (b) For all $s \in S$, compute
$$v'(s) \leftarrow \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}. \tag{5.108}$$
   (c) $\sigma \leftarrow \text{sp}(\mathbf{v}' - \mathbf{v})$.

3. **Terminate:** For all $s \in S$, return
$$d_\epsilon(s) \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}$$
and
$$v_\lambda^\epsilon(s) = v'(s) + \frac{\lambda}{1-\lambda}(\underline{\mathbf{v}' - \mathbf{v}}).$$

Before providing a proof of convergence, some comments are in order:

1. The span (as opposed to the norm) is used as the basis for terminating the algorithm. Section 5.5.1 suggested that the span-based stopping criterion will be met in fewer iterations than the norm-based stopping criterion. This will be explored in numerical examples below.

2. The condition $\sigma \geq (1-\lambda)\epsilon/\lambda$ will be shown to result in an ϵ-optimal policy upon termination. This should be sufficient in most practical examples.

3. The expression for $v^\epsilon(s)$, referred to as the lower bound extrapolation in Section 5.5.4, can be determined with little extra computational effort. It is important to emphasize that $\mathbf{v}^\epsilon$ should be used instead of $\mathbf{v}$ to approximate $\mathbf{v}_\lambda^*$ because Theorem 5.13 ensures that each component of the lower bound extrapolation will be within ϵ of the optimal value function.

4. The iterative step (5.108) represents the essence of the algorithm. In operator notation, it was represented by $\mathbf{v}' = L\mathbf{v}$.

5. Judicious choice of a starting value can enhance convergence. Choosing the lower bound $v(s) = (1-\lambda)^{-1} \min_{a \in A_s} r(s,a)$ ensures monotone convergence.

6. In most practical examples, the transition probabilities will be mostly zero so that for each $s \in S$, the sum on the right hand side of (5.108) will only contain a

few terms. For example, in the queuing service rate control model (Section 3.4.1), there will be at most three non-zero probabilities in each state, enabling the sum to be easily coded and quickly computed.

7. When the stopping criterion is satisfied, specifying $d_\epsilon(s)$ requires no addition computation. It can be determined while computing (5.108). Theorem 5.13 ensures that it is ϵ-optimal. Note that its value is not determined by the algorithm and $d_\epsilon(s)$ need not be unique.

8. Action elimination can be used to enhance the algorithm. This will be discussed below.

9. In addition to computing ϵ-optimal policies, value iteration can be used to demonstrate the structure of optimal policies.

10. Some authors refer to the above algorithm as *synchronous* value iteration because at each iteration, the estimate is updated in all states. Gauss-Seidel value iteration (described below) provides an *asynchronous* alternative.

The following theorem establishes convergence of value iteration to an ϵ-optimal policy. Its proof follows by combining several of the above theorems.

Theorem 5.16. Suppose $0 \leq \lambda < 1$ and $\epsilon > 0$. Then for any $\mathbf{v}^0 \in V$ the sequence $\mathbf{v}^n, n = 1, 2, \ldots$ generated by value iteration has the following properties:

1. $\|\mathbf{v}^n - \mathbf{v}^*_\lambda\| \to 0$ as $n \to \infty$.
2. $\mathrm{sp}(\mathbf{v}^n - \mathbf{v}^*_\lambda) \to 0$.
3. For each $\epsilon > 0$, there exists a finite N which for $n \geq N$, (5.107) holds.
4. When (5.107) is satisfied,
   (a) $\|\mathbf{v}^\epsilon - \mathbf{v}^*_\lambda\| < \epsilon$, and
   (b) $\|\mathbf{v}_\lambda^{d_\epsilon^\infty} - \mathbf{v}^*_\lambda\| < \epsilon$.

Proof. Proposition 5.1 establishes that L is a contraction mapping. Hence by Theorem 5.5, $\mathbf{v}^n$ converges to a fixed point of L in norm. From Theorem 5.8, the limit is unique and equals $\mathbf{v}^*_\lambda$. Since $\mathrm{sp}(\mathbf{v}) \leq 2\|\mathbf{v}\|$, part 2 follows. Part 3 follows from Theorem 5.11 and the observation that

$$\mathrm{sp}(\mathbf{v}^{n+1} - \mathbf{v}^n) = \mathrm{sp}(L^{n+1}\mathbf{v}^0 - L^n\mathbf{v}^0) \leq (\lambda\gamma^*)^n \mathrm{sp}(L\mathbf{v}^0 - \mathbf{v}^0).$$

Part 4 is a consequence of Theorem 5.13. □

Example 5.11. This example finds an ϵ-optimal policy in the two-state model using value iteration with $\lambda = 0.9$, $\epsilon = 0.000001$ and starting value $\mathbf{v} = \mathbf{0}$.

Using $\|\mathbf{v}' - \mathbf{v}\|$, the algorithm terminates after 162 iterations and using $\text{sp}(\mathbf{v}' - \mathbf{v})$, the algorithm terminates after 16 iterations. In both cases the algorithm terminates with the *optimal* stationary policy that uses decision rule $d_\epsilon = (a_{1,2}, a_{2,2,})$.

Figure 5.6 compares the speed of convergence of $\mathbf{v}' - \mathbf{v}$ with respect to these two measures. Observe that the span-based criterion converges more quickly than the norm-based criterion for the same sequence of iterates. This in a consequence of the observation in Example 5.8, that $\gamma^* = 0.8$.

At termination using the span stopping criterion, $\mathbf{v} = (25.39, 23.19)$ and $\mathbf{v}^\epsilon = (30.15, 27.94)$. Using the norm stopping criterion, $\mathbf{v} = \mathbf{v}^\epsilon = (30.15, 27.94)$. Thus the span converges 10 times faster than the norm. Moreover the lower bound extrapolation produces identical estimates of the optimal value function at termination under either a norm-based or span-based stopping criterion.

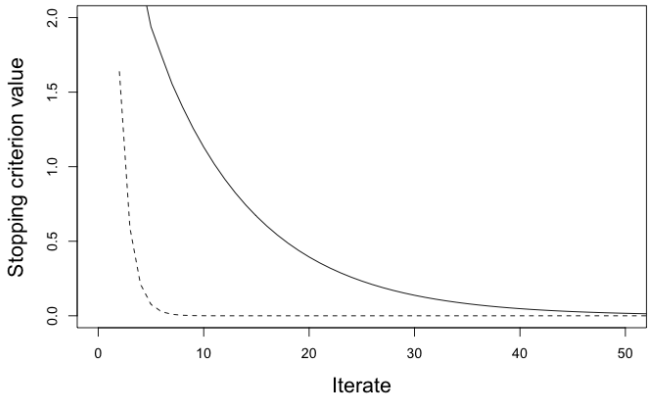


Figure 5.6: Comparison of convergence rate of $\text{sp}(\mathbf{v}' - \mathbf{v})$ (dashed line) to $||\mathbf{v}' - \mathbf{v}||$ (solid line) in two-state example.

Convergence rates and error bounds

Algorithms are frequently compared on the basis of their convergence rates. A discussion of some relevant concepts follows.

Let $\mathbf{u}^n; n = 1, 2, \ldots$ denote a sequence of vectors converging (in norm) to $\mathbf{u}^*$. Then $\mathbf{u}^n$ *converges with order* $\alpha > 0$ if there exists a constant $K > 0$ for which

$$||\mathbf{u}^{n+1} - \mathbf{u}^*|| \leq K||\mathbf{u}^n - \mathbf{u}^*||^\alpha.$$

Convergence of order 1 is referred to as *linear convergence* and convergence with order

2 is referred to as *quadratic convergence*[23]. Clearly for fixed K, the higher the order, the faster the convergence[24].

The sequence $\mathbf{u}^n, n = 1, 2, \ldots$ *converges geometrically at rate* β if there exists a constant $k > 0$ and $0 < \beta < 1$ for which

$$||\mathbf{u}^n - \mathbf{u}^*|| \leq K\beta^n.$$

This is often expressed as $\mathbf{u}^n = \mathbf{u}^* + O(\beta^n)$.

The following result summarizes convergence rate properties of value iteration.

Theorem 5.17. Let $\mathbf{v}^n : n = 1, 2, \ldots$ denote a sequence of iterates of value iteration where $\mathbf{v}^0$ is arbitrary. Then:

1. $\mathbf{v}^n$ converges linearly to $\mathbf{v}^*_\lambda$.
2. $\mathbf{v}^n$ converges to $\mathbf{v}^*_\lambda$ at rate λ^n.
3. For $n = 1, 2, \ldots$

$$||\mathbf{v}^{n+1} - \mathbf{v}^*_\lambda|| = \frac{\lambda^n}{(1-\lambda)}||\mathbf{v}^1 - \mathbf{v}^0|| \tag{5.109}$$

Proof. Parts 1 and 2 follow directly from the definitions. To prove part 3, note that

$$\begin{aligned}
\|\mathbf{v}^{n+1} - \mathbf{v}^*_\lambda\| &= \|L\mathbf{v}^n - L\mathbf{v}^*_\lambda\| \\
&\leq \|L\mathbf{v}^n - L\mathbf{v}^{n+1}\| + \|L\mathbf{v}^{n+1} - L\mathbf{v}^*_\lambda\| \\
&\leq \|L^n\mathbf{v}^1 - L^n\mathbf{v}^0\| + \lambda\|\mathbf{v}^{n+1} - \mathbf{v}^*_\lambda\| \\
&\leq \lambda^n\|\mathbf{v}^1 - \mathbf{v}^0\| + \lambda\|\mathbf{v}^{n+1} - \mathbf{v}^*_\lambda\|.
\end{aligned}$$

The result follows by rearranging terms in the last inequality. □

Expression (5.109) is referred to as an *error bound* and is consistent with the first two statements in the theorem.

5.6.2 Gauss-Seidel value iteration

Gauss-Seidel is a variant of value iteration that updates values as soon as they have been computed instead of waiting until finishing a complete pass through step 2 of Algorithm 5.3. Its motivation is that using updated values as soon as possible will speed up convergence. Proving that this is the case requires machinery that is outside the scope of this book but numerical results below support this conjecture.

[23] For example when solving $f(x) = x$ or equivalently $f(x) - x = 0$, under mild conditions, successive approximations converges linearly and Newton's method converges quadratically.

[24] Note that when $\alpha > 1$, this statement only makes when the error is less than 1.

A description of a Gauss-Seidel value iteration algorithm follows. Assume that value iteration evaluates states using (5.108) in order $s_1, s_2, \ldots, s_M$. Noting that $v'(s_j)$ for $j < k$ have been calculated prior to evaluating $v'(s_k)$, Gauss-Seidel value iteration accounts for this by replacing (5.108) in state s_k with

$$v'(s_k) = \max_{a \in A_{s_k}} \left\{ r(s_k, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) v'(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v(s_j) \right) \right\}. \quad (5.110)$$

Thus, it uses the most current estimate of the value function in all states when doing its update. It is most straightforward to state in component notation.

For $k = 1, \ldots, M$, inductively define the s_k-th component of the operator $G : V \to V$ by

$$G\mathbf{v}(s_k) := \max_{a \in A_{s_k}} \left\{ r(s_k, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) G\mathbf{v}(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v(s_j) \right) \right\}. \quad (5.111)$$

This means that $G\mathbf{v}(s_1) = L\mathbf{v}(s_1)$,

$$G\mathbf{v}(s_2) := \max_{a \in A_{s_2}} \left\{ r(s_2, a) + \lambda \left(p(s_1|s_2, a) G\mathbf{v}(s_1) + \sum_{j=2}^{M} p(s_j|s_2, a) v(s_j) \right) \right\},$$

$$G\mathbf{v}(s_3) := \max_{a \in A_{s_3}} \left\{ r(s_3, a) + \lambda \left(\sum_{j=1}^{2} p(s_j|s_3, a) G\mathbf{v}(s_j) + \sum_{j=3}^{M} p(s_j|s_3, a) v(s_j) \right) \right\}$$

and so forth.

An iterative algorithm that uses Gauss-Seidel updates follows. It assumes that states are labeled $s_1, \ldots, s_M$ and are evaluated in that order. It is clearest when stated in component form.

Algorithm 5.4. Gauss-Seidel value iteration: component form

1. **Initialize:**
   (a) Specify $v'(s)$ for all $s \in S$.
   (b) Specify $\epsilon > 0$ and $\sigma > (1 - \lambda)\epsilon/2\lambda$
2. **Iterate:** While
$$\sigma \geq \frac{(1 - \lambda)}{2\lambda} \epsilon : \quad (5.112)$$
   (a) $v(s) \leftarrow v'(s)$ for all $s \in S$.

(b) For $k = 1, \ldots, M$, compute

$$v'(s_k) \leftarrow \max_{a \in A_s} \left\{ r(s,a) + \lambda \left(\sum_{j<k} p(s_j|s_k,a) v'(s_j) + \sum_{j \geq k} p(s_j|s_k,a) v(s_j) \right) \right\}. \tag{5.113}$$

(c) $\sigma \leftarrow \|\mathbf{v}' - \mathbf{v}\|$.

3. **Terminate:** For $k = 1, \ldots, M$, return

$$d_\epsilon(s_k) \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \left(\sum_{j<k} p(s_j|s_k,a) v'(s_j) + \sum_{j \geq k} p(s_j|s_k,a) v(s_j) \right) \right\}. \tag{5.114}$$

Some comments about the algorithm follow.

1. This update is said to be *asynchronous* because state values are updated prior to completing a pass through an iteration. Online reinforcement learning and simulation-based algorithms (Chapters 10 and 11) use asynchronous updates since values are updated as soon as states are observed or simulated.

2. This algorithm uses a norm-based stopping criterion since the bounds used to derive the span-based criterion for value iteration are not available. If one wishes to instead use a span-based stopping criterion, it can be applied intermittently after using a value iteration update.

3. The decision rule specification in step 3 is consistent with the key iterative step based on (5.113). Doing so simplifies the proof that $\|\mathbf{v}^{(d_\epsilon)^\infty} - \mathbf{v}^*_\lambda\| < \epsilon$ at termination. Alternatively one can choose d_ϵ to be $\mathbf{v}'$-greedy (corresponding to an extra value iteration step)

4. The ordering of the states is immaterial and can vary from iteration to iteration. The analysis in this section remains valid as long as there is a complete pass through all states at each iteration.

5. When the transition matrices corresponding to all decision rules are upper-triangular, Gauss-Seidel value iteration and value iteration are identical.

Convergence of Gauss-Seidel iteration

A proof of the convergence of Gauss-Seidel iteration to $\mathbf{v}^*_\lambda$ is based on showing that the operator representing a Gauss-Seidel update is a contraction mapping and then applying the Banach fixed point theorem. The following result is obtained by modifying the proof of Proposition 5.1 accordingly.

Proposition 5.8. Suppose $G : V \to V$ is defined by (5.111) and $0 \leq \lambda < 1$. Then G is a contraction mapping with modulus (at most) λ with respect to the sup-norm.

Proof. The proof uses induction over states $s \in S$. Choose $\mathbf{v}$ and $\mathbf{u}$ in V. For $s = s_1$, since $G\mathbf{v}(s_1) = L\mathbf{v}(s_1)$, it follows that

$$|G\mathbf{v}(s_1) - G\mathbf{u}(s_1)| \leq \lambda \|\mathbf{v} - \mathbf{u}\| \tag{5.115}$$

from the same argument used in the proof of Proposition 5.1.

Next assume (5.115) holds when s_j replaces s_1 for $j = 1, \ldots, k-1$. Define $a' \in A_{s_k}$ by

$$\begin{aligned} &r(s_k, a') + \lambda \left(\sum_{j<k} p(s_j|s_k, a') G\mathbf{v}(s_j) + \sum_{j \geq k} p(s_j|s_k, a') v(s_j) \right) \\ &= \max_{a \in A_s} \left\{ r(s_k, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) G\mathbf{v}(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v(s_j) \right) \right\} = G\mathbf{v}(s_k). \end{aligned}$$

Assuming first that $G\mathbf{v}(s_k) \geq G\mathbf{u}(s_k)$,

$$\begin{aligned} 0 &\leq G\mathbf{v}(s_k) - G\mathbf{u}(s_k) \\ &\leq r(s_k, a') + \lambda \left(\sum_{j<k} p(s_j|s_k, a') G\mathbf{v}(s_j) + \sum_{j \geq k} p(s_j|s_k, a') v(s_j) \right) \\ &\quad - \left\{ r(s_k, a') + \lambda \left(\sum_{j<k} p(s_j|s_k, a') G\mathbf{u}(s_j) + \sum_{j \geq k} p(s_j|s_k, a') u(s_j) \right) \right\} \\ &= \lambda \left(\sum_{j<k} p(j|s_k, a') \Big(G\mathbf{v}(s_j) - G\mathbf{u}(s_j) \Big) + \sum_{j \geq k} p(j|s_k, a') \Big(v(s_j) - u(s_j) \Big) \right) \\ &\leq \lambda \left(\sum_{j<k} p(s_j|s_k, a') \big| G\mathbf{v}(s_j) - G\mathbf{u}(s_j) \big| + \sum_{s_j \geq k} p(j|s_k, a') \|\mathbf{v} - \mathbf{u}\| \right) \\ &\leq \lambda \|\mathbf{v} - \mathbf{u}\|, \end{aligned}$$

where the last inequality follows from the induction hypothesis.

Repeating the above argument for $G\mathbf{u}(s_k) \geq G\mathbf{v}(s_k)$ establishes that (5.115) holds for s_k, completing the induction step and the proof. □

The following theorem establishes convergence properties of Gauss-Seidel iteration. The proof of parts 2 and 3 differ from that for value iteration since bounds like those in (5.93) are not available for $G\mathbf{v}$.

Theorem 5.18. Suppose $0 \leq \lambda < 1$ and $\epsilon > 0$. Let the sequence $\mathbf{v}^n, n = 1, 2, \ldots$ be generated by Gauss-Siedel value iteration. Then:

1. $\|\mathbf{v}^n - \mathbf{v}^*_\lambda\| \to 0$ as $n \to \infty$,
2. $\text{sp}(\mathbf{v}^n - \mathbf{v}^*_\lambda) \to 0$.
3. For each $\epsilon > 0$, there exists a finite N which for $n \geq N$, (5.112) holds.
4. When (5.112) is satisfied,
   (a) d^∞_ϵ is ϵ-optimal,
   (b) for $n \geq N$, $\|\mathbf{v}^n - \mathbf{v}^*_\lambda\| < \epsilon/2\lambda$ and $\text{sp}(\mathbf{v}^n - \mathbf{v}^*_\lambda) < \epsilon/\lambda$.

Proof. Since G is a contraction, $G^n\mathbf{v}_0$ converges to a fixed point $\mathbf{v}^*$ by Theorem 5.5. Since $G\mathbf{v}^* = \mathbf{v}^*$, from (5.111),

$$v^*(s_k) = G\mathbf{v}^*(s_k) = \max_{a \in A_s} \left\{ r(s_k, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) G\mathbf{v}^*(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v^*(s_j) \right) \right\}$$
$$= \max_{a \in A_s} \left\{ r(s_k, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) v^*(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v^*(s_j) \right) \right\} = L\mathbf{v}^*(s_k)$$

Hence, $\mathbf{v}^*$ is also a fixed point of L. Since L has the unique fixed point $\mathbf{v}^*_\lambda$ the first result follows. Part 2 follows by noting again that $\text{sp}(\mathbf{v}) \leq 2\|\mathbf{v}\|$. Part 3 follows directly from part 1.

To prove 4(a), note that in Step 4 of Algorithm 5.4 that $G\mathbf{v}^{n+1} = G_{d_\epsilon}\mathbf{v}^{n+1}$ where G_d denotes the operator that corresponds to implementing a single step of Gauss-Seidel value iteration with a fixed decision rule d. Hence

$$\begin{aligned} \|\mathbf{v}_\lambda^{(d_\epsilon)^\infty} - \mathbf{v}^*_\lambda\| &\leq \|\mathbf{v}_\lambda^{(d_\epsilon)^\infty} - G_{d_\epsilon}\mathbf{v}^n\| + \|G\mathbf{v}^n - \mathbf{v}^*_\lambda\| \\ &\leq \lambda\|\mathbf{v}^{(d_\epsilon)^\infty} - \mathbf{v}^n\| + \lambda\|\mathbf{v}^n - \mathbf{v}^*_\lambda\|, \end{aligned} \tag{5.116}$$

where the last inequality follows by noting that G and G_{d_ϵ} are contraction mappings with fixed points $\mathbf{v}^*_\lambda$ and $\mathbf{v}_\lambda^{(d_\epsilon)^\infty}$, respectively. It is easy to show[25] that each of the quantities on the right hand side of (5.116) is bounded by $(1-\lambda)^{-1}\|\mathbf{v}^{n+1} - \mathbf{v}^n\|$, from which the result follows.

To prove 4(b), since $\mathbf{v}^{n+1} = G\mathbf{v}^n$ and $\mathbf{v}^*_\lambda$ is the fixed point of G,

$$\begin{aligned} \|\mathbf{v}^n - \mathbf{v}^*_\lambda\| &\leq \|\mathbf{v}^{n+1} - \mathbf{v}^n\| + \|\mathbf{v}^{n+1} - \mathbf{v}^*_\lambda\| \\ &\leq \|\mathbf{v}^{n+1} - \mathbf{v}^n\| + \|G\mathbf{v}^n - G\mathbf{v}^*_\lambda\| \leq \|\mathbf{v}^{n+1} - \mathbf{v}^n\| + \lambda\|\mathbf{v}^n - \mathbf{v}^*_\lambda\|. \end{aligned}$$

Hence $(1-\lambda)\|\mathbf{v}^n - \mathbf{v}^*_\lambda\| \leq \|\mathbf{v}^{n+1} - \mathbf{v}^n\|$, which gives the result. □

[25] See Puterman (1994) p. 162 or apply a telescoping sum argument.

Example 5.12. This example applies Gauss-Seidel value iteration to the two-state model starting from $\mathbf{v} = \mathbf{0}$ and setting $\lambda = 0.9$ and $\epsilon = 0.000001$.

Using stopping criterion (5.112), the algorithm terminates in 120 iterations with $\mathbf{v}' = (30.147, 27.941)$. Note that $\mathbf{v}' = \mathbf{v}^{(d_\epsilon)^\infty} = \mathbf{v}^*_\lambda$ (to at least 3 decimal places) so that the ϵ-optimal policy is optimal.

Figure 5.7 compares the speed of convergence of $\|\mathbf{v}' - \mathbf{v}'\|$ for value iteration and Gauss-Seidel value iteration. Note that for successive iterates $\mathbf{v}'', \mathbf{v}'$ and $\mathbf{v}$, the ratio $\|\mathbf{v}'' - \mathbf{v}'\| / \|\mathbf{v}' - \mathbf{v}\| \approx 0.86$ for Gauss-Seidel, while equaling 0.9 for value iteration. This suggests a smaller contraction modulus for Gauss-Seidel in this example.

The figure also shows $\text{sp}(\mathbf{v}' - \mathbf{v})$ for Gauss-Seidel indicating faster convergence with respect to this measure, even in the absence of theoretical justification.

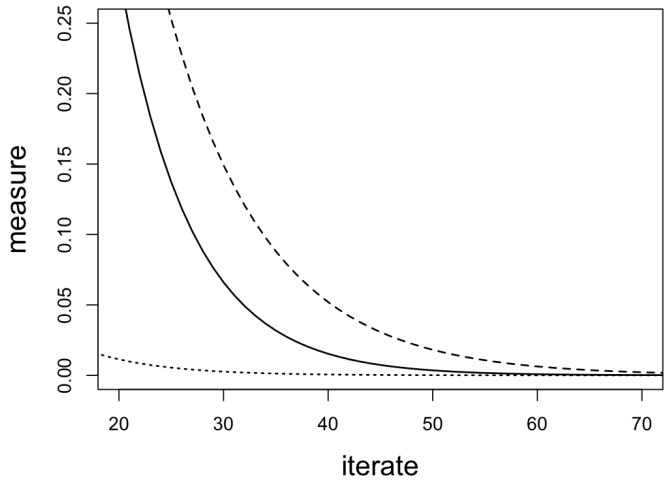


Figure 5.7: Comparison of speed of convergence of Gauss-Seidel and value iteration for 2-state example. The solid line shows $\|\mathbf{v}' - \mathbf{v}\|$ for Gauss-Seidel, the dashed line shows $\|\mathbf{v}' - \mathbf{v}\|$ for value iteration and the dotted line shows $\text{sp}(\mathbf{v}' - \mathbf{v})$ for Gauss-Seidel.

A hybrid approach plus some discussion

Appendix 5.13.2 provides bounds and develops some interesting matrix representations for Gauss-Seidel iteration. As shown in Section 5.5.4, the span stopping criterion for value iteration is based on subtracting the lower bound from the upper bound as in Theorem 5.13. Since the lower bound for Gauss-Seidel cannot be evaluated easily, a hybrid algorithm that intersperses a value iteration step in Gauss-Seidel iteration

enables computation of bounds and allows use of the span-based criterion. Hence, we recommend in practice to either:

1. use such a hybrid approach when solving large problems with Gauss-Seidel, or
2. simply use value iteration with the span stopping criterion.

The advantage of the later approach is the availability of bounds, the applicability of the span-based stopping criterion and termination with a good estimate of the value function. The advantage of the former approach is that in large problems, updates of the value function can be used more quickly, perhaps leading to faster convergence.

Asynchronous value iteration

In an asynchronous value iteration algorithm states are updated one at a time. In particular a single update is given by

$$v^{n+1}(s) = \begin{cases} \max_{a \in A_s} \{r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v^n(j)\} & s = s' \\ v^n(s) & s \neq s'. \end{cases} \tag{5.117}$$

for some chosen state s'. If all states are cycled through without repeating any states, this is exactly a Gauss-Seidel update. However, if states are chosen randomly, some might repeat before others are evaluated even once. As long as all states are evaluated infinitely often, the algorithm converges to $\mathbf{v}^*_\lambda$.

Asynchronous updates underlie simulation-based algorithms discussed in Chapter 10 and also updates generated by parallel computation. The reader is referred to the literature for more details and a proof of convergence.

5.6.3 Value iteration with action elimination

The purpose of augmenting an algorithm with an action elimination procedure is to reduce the number of actions evaluated when performing a maximization. This brief section shows how to use the result in Proposition 5.7 to do so in the context of value iteration. It requires minimal extra computation but some additional storage.

Algorithm 5.5. Value iteration with action elimination: component form

1. **Initialize:**
   (a) Specify $v'(s)$ for all $s \in S$.
   (b) Set $H_s \leftarrow A_s$ for all $s \in S$.
   (c) Specify $\epsilon > 0$ and $\sigma > (1-\lambda)\epsilon/\lambda$.

2. **Iterate:** While $\sigma \geq (1-\lambda)\epsilon/\lambda$ and H_s contains two or more elements for some $s \in S$:

   (a) $v(s) \leftarrow v'(s)$ for all $s \in S$.

   (b) **Update values:** For all $s \in S$ and $a' \in H_s$, compute

   $$q(s,a') \leftarrow r(s,a') + \lambda \sum_{j \in S} p(j|s,a')v(j) \tag{5.118}$$

   and

   $$v'(s) \leftarrow L\mathbf{v}(s) = \max_{a \in H_s} q(s,a).$$

   (c) **Identify and eliminate non-optimal actions:** For all $s \in S$,

   $$H_s \leftarrow \left\{ a' \in H_s \;\middle|\; \frac{\lambda}{(1-\lambda)} \mathrm{sp}(\mathbf{v}' - \mathbf{v}) \geq v'(s) - q(s,a') \right\}. \tag{5.119}$$

   (d) $\sigma \leftarrow \mathrm{sp}(\mathbf{v}' - \mathbf{v})$.

3. **Terminate:** For all $s \in S$, return

   $$d_\epsilon(s) \in \arg\max_{a \in H_s} q(s,a) \tag{5.120}$$

   and

   $$v_\lambda^\epsilon(s) = v'(s) + \frac{\lambda}{1-\lambda}(\underline{\mathbf{v}' - \mathbf{v}}). \tag{5.121}$$

Some comments on this algorithm follow:

1. Applying this algorithm results in *either*:

   (a) an ϵ-optimal policy when terminated with the span-based stopping criterion, or

   (b) the unique optimal policy when H_s contains a single element for all $s \in S$.

   In both cases $\mathbf{v}^\epsilon$ is within ϵ of the optimal value function.

2. The algorithm identifies an optimal policy, rather than an ϵ-optimal policy, only when it terminates when H_s contains a single element for all $s \in S$.

3. Note that it is necessary to evaluate $L\mathbf{v}(s)$ for all $s \in S$ before eliminating actions. This requires steps 2(b) and 2(c).

4. The quantity $q(s,a)$ does not need to be stored from iteration to iteration. Its evaluation requires no extra computation at each iteration since it is an interme-

diate calculation in determining $L\mathbf{v}$. When using value iteration without action elimination, it need not be stored for any state.

5. Note that $|H_s|$ is non-increasing for all $s \in S$ between successive iterations.

6. This algorithm does not generalize easily to to Gauss-Seidel value iteration because of the unavailability of easily computable lower bounds.

Example 5.10 shows that starting with $\mathbf{v}^0 = (5, -5)$, action $a_{2,1}$ is eliminated in s_1 at the third iteration and action $a_{1,1}$ is eliminated in state s_1 after the sixth iteration. Thus it requires only **six** iterations to identify the optimal policy using action elimination in the two-state example. Such dramatic results cannot be expected in all applications.

5.7 Policy iteration

Policy iteration (also known as policy improvement or approximation in policy space) offers an attractive alternative to value iteration for finding optimal policies in discounted Markov decision processes. Actor-critic methods (see Chapter 11) may be regarded as policy iteration using function approximation and simulation.

While value iteration has its foundations in fixed point theory, policy iteration exploits the special structure of Markov decision processes, but may be also viewed as an application of Newton's method for finding the zero of a function. While value iteration may only find an ϵ-optimal policy, policy iteration will find an optimal policy in a finite state and action model.

5.7.1 The policy iteration algorithm

A schematic representation of policy iteration appears in Figure 5.8. It shows that policy iteration cycles between an evaluation step that solves a linear system to find the value of a policy and an improvement step that chooses a greedy decision rule with respect to the value of the previously evaluated policy. When greedy decision rules repeat, which occurs with certainty in a finite state and action model, the algorithm terminates with an optimal stationary deterministic policy.

A statement of policy iteration in vector notation follows. We recommend that you implement the component form of the algorithm stated below, because it makes the improvement step more explicit.

Algorithm 5.6. Policy iteration: vector form

1. **Initialize:** Specify $d \in D^{\text{MD}}$ and set $\Gamma \leftarrow S$.

2. **Iterate:** While $\Gamma \neq \emptyset$:

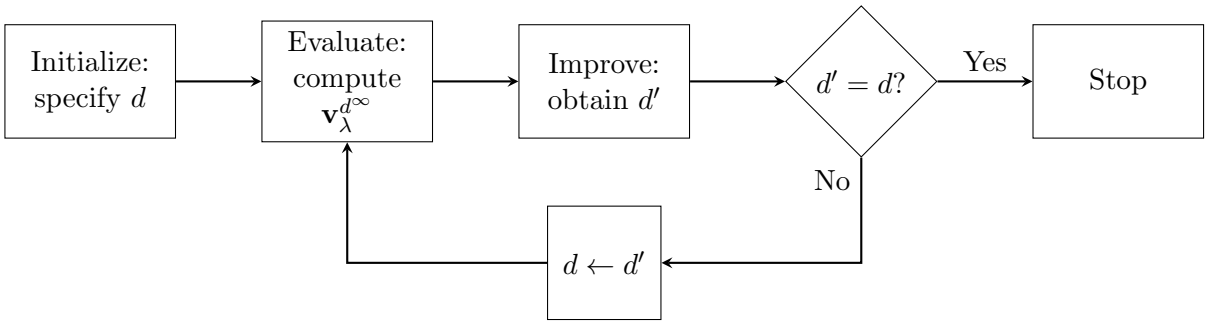


Figure 5.8: Flow diagram for policy iteration

(a) **Evaluate:** Find $\mathbf{v}'$ by solving

$$(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{r}_d. \tag{5.122}$$

(b) **Improve:** Choose

$$d' \in \underset{\delta \in D^{\mathrm{MD}}}{\arg\,\text{c-max}} \left\{ \mathbf{r}_\delta + \lambda \mathbf{P}_\delta \mathbf{v}' \right\}, \tag{5.123}$$

setting $d' = d$ if possible.

(c) $\Gamma \leftarrow \{s \in S \,|\, d'(s) \neq d(s)\}$.

(d) **Update:** $d \leftarrow d'$.

3. **Terminate:** Return $d^* = d$ and $\mathbf{v}_\lambda^* = \mathbf{v}'$.

Before stating the algorithm in a form that is more suitable for computation, note that:

1. In contrast to value iteration, policy iteration begins with a Markovian deterministic decision rule instead of a value. On the other hand, the algorithm can start in step 2(b) with any $\mathbf{v} \in V$.

2. The evaluation step involves solving a system of linear equations, which requires at most $O(|S|^3)$ operations. Modified policy iteration, described below, replaces this step with an iterative method for approximating the value function of d^∞.

3. The improvement step requires the same effort as one pass through value iteration. We emphasize that it is implemented component-wise thus avoiding enumerating all Markovian deterministic decision rules.

4. The instruction "set $d' = d$ if possible" prevents cycling when the maximizer is non-unique in some states. It is also ensures that algorithm terminates in a

finite number of iterations when there are finitely many deterministic stationary policies. This rule can be replaced by a stopping criterion expressed in terms of value functions: stop when successive value functions are sufficiently close. The advantage of such a rule is that it can be used to terminate the algorithm when the set of states and/or actions is not finite.

5. An action elimination procedure can be incorporated into policy iteration.

6. The next section shows that policy iteration can be implemented asynchronously. That is, as soon as a strict improvement is identified in a state or subset of states, the improved policy can be updated.

A statement of policy iteration in component notation follows. This description is more suitable for computation and emphasizes that the improvement step should be implemented state by state.

Algorithm 5.7. Policy iteration: component form

1. **Initialize:** Specify $a_s \in A_s$ for all $s \in S$ and set $\Gamma \leftarrow S$.

2. **Iterate:** While $\Gamma \neq \emptyset$:

   (a) **Evaluate:** Obtain $v'(s)$ for all $s \in S$ by solving the system of linear equations

   $$v(s) - \lambda \sum_{j \in S} p(j|s, a_s) v(j) = r(s, a_s). \tag{5.124}$$

   (b) **Improve:** For all $s \in S$, choose

   $$a'_s \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v'(j) \right\}, \tag{5.125}$$

   setting $a'_s = a_s$ if possible.

   (c) $\Gamma \leftarrow \{s \in S \,|\, a'_s \neq a_s\}$.

   (d) **Update:** For all $s \in S$, $a_s \leftarrow a'_s$.

3. **Terminate:** Return $d^*(s) = a_s$ and $v^*_\lambda(s) = v'(s)$ for all $s \in S$.

An alternative (asynchronous) form of the algorithm jumps immediately to the evaluation step as soon as there is a strict improvement in some state in step 2(b).

That is for some s and $\delta > 0$

$$\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\} = r(s,a_s) + \lambda \sum_{j \in S} p(j|s,a_s) v(j) + \delta. \tag{5.126}$$

The proof of the theorem below also establishes that this variant of policy iteration terminates in a finite number of iterations.

Examples

The following example applies Algorithm 5.7 to the two-state model.

Example 5.13. Let $\lambda = 0.9$. The algorithm proceeds as follows:

1. Choose $a_{s_1} = a_{1,2}$ and $a_{s_2} = a_{2,1}$.
2. Evaluate[a] the stationary policy that use these actions to obtain $v(s_1) = -40$ and $v(s_2) = -50$.
3. In the improvement step

$$\begin{aligned} a'_{s_1} &= \arg\max_{i=1,2} \left\{ r(s_1, a_{1,i}) + \lambda \sum_{j=1,2} p(s_j|s_1, a_{1,i}) v(s_j) \right\} \\ &= \arg\max\{3 + 0.9\big(0.8v(s_1) + 0.2v(s_2)\big), 5 + 0.9v(s_2)\} \\ &= \arg\max\{-34.8, -40\} = a_{1,1} \end{aligned}$$

   and

$$\begin{aligned} a'_{s_2} &= \arg\max_{i=1,2} \left\{ r(s_2, a_{2,i}) + \lambda \sum_{j=1,2} p(s_j|s_2, a_{2,i}) v(s_j) \right\} \\ &= \arg\max\{-5 + 0.9v(s_2), 2 + 0.9\big(0.4v'(s_1) + 0.6v(s_2)\big)\} \\ &= \arg\max\{-50, -39.4\} = a_{2,2} \end{aligned}$$

4. Since $a'_s \neq a_s$ for all $s \in S$, replace a_s by a'_s for all $s \in S$ and return to the evaluation step.
5. Appealing again to Example 5.39, shows that $v(s_1) = 27.188$ and $v(s_2) = 25.625$.
6. At the next improvement step,

$$\begin{aligned} a'_{s_1} &= \arg\max\{27.188, 28.063\} = a_{1,2} \\ a'_{s_2} &= \arg\max\{18.063, 25.625\} = a_{2,2}. \end{aligned}$$

7. Since $a'_s \neq a_s$ for all $s \in S$, replace a_s by a'_s for all $s \in S$ and return to the evaluation step.

8. From Example 5.39, $v(s_1) = 30.147$ and $v(s_2) = 27.941$.

9. Applying the improvement step gives

$$a'_{s_1} = \arg\max\{29.735, 30.147\} = a_{1,2}$$
$$a'_{s_2} = \arg\max\{20.146, 27.941\} = a_{2,2}.$$

10. Since $a'_s = a_s$ for all $s \in S$, stop.

The above calculations showed that it required two improvement steps to find the optimal policy and an additional improvement step to confirm that it was optimal. Using action elimination may have avoided this additional step. Observe that the algorithm terminates with:

1. the optimal policy,
2. its value function, and
3. the solution of the Bellman equation.

Observe also that the sequence of value functions is non-decreasing.

[a]Note that Example 5.4 evaluates all four stationary deterministic policies.

As illustrated in the example above, it usually requires only a few iterations to find an optimal policy. However, the following example[26] shows that policy iteration may require $|S| + 1$ iterations to find an optimal policy.

Example 5.14. Let $S = \{1, 2, \ldots, M, \Delta\}$, $A_s = \{a_0, a_1\}$, $s \leq M$ and $A_\Delta = \{a_0\}$ where a_0 represents "stay" and a_1 represents "go right". The reward function is $r(s, a_0) = 0,\ s \in S$, $r(s, a_1) = -1,\ s < M$, and $r(M, a_1) = M^2$. The transition probabilities are $p(s|s, a_0) = 1$ for $s \in S$ and $p(s + 1|s, a_1) = 1$ for $s < M$. See Figure 5.9.

Implement policy iteration as follows. Initialize by choosing $a_s = a_0$ for all $s \in S$. Evaluation shows that $v(s) = 0$ for all $s \in S$. In the improvement step, $a'_s = a_0$ for $s \neq M$ and $a'_M = a_1$. Since $a'_s \neq a_s$ for all $s \in S$, replace a_s by a'_s for

[26]John Tsitsiklis provided this example in a personal communication. A different example reaching the same conclusion appears in Bertsekas [2012].

all $s \in S$ and return to the start of step 2 in the algorithm.

Evaluating this policy shows that $v(s) = 0$ for $s < M$ and $v(M) = M^2$. At the next improvement step $a'_s = a_1$ for $s \in \{M-1, M\}$ and $a'_s = a_0$ otherwise. Evaluating this policy shows that

$$v(s) = \begin{cases} 0 & s \notin \{M-1, M\} \\ -1 + \lambda M^2 & s = M-1 \\ M^2 & s = M. \end{cases}$$

Continuing in this way shows that it requires $M+1$ iterations to identify an optimal policy. The reason for this slow convergence is that it takes M iterations for the reward of M^2 to "make its way" back to state 1. Note that this example is representative of games like chess where the reward corresponding to winning, losing or ending in a draw may require many transitions to be realized.

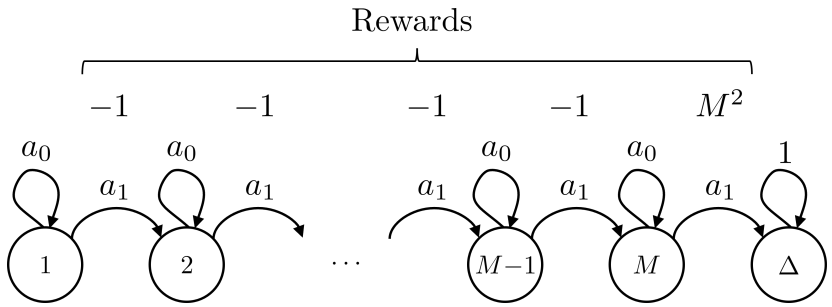


Figure 5.9: Schematic representation of model in Example 5.14. Actions a_0 and a_1 indicate transitions with probability 1 given the chosen action. Rewards are associated with transitions to the "right".

5.7.2 Convergence of policy iteration

This section formally establishes the result that policy iteration finds an optimal policy in a finite number of iterations, assuming a finite state and action model. It begins by showing that the sequence of values generated by policy iteration is non-decreasing and moreover a *strict* improvement leads to a strict increase in value. Vector notation simplifies exposition.

Proposition 5.9. Suppose $\mathbf{v}$ is the solution of $(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{r}_d$ and for some $s' \in S$ and $\delta > 0$

$$\mathbf{r}_{d'} + \lambda \mathbf{P}_{d'}\mathbf{v} = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} + \delta \mathbf{e}_{s'}, \tag{5.127}$$

where $\mathbf{e}_{s'}$ denotes a vector with a one in the s'-th position and zeroes in all other components. Then

$$\mathbf{v}_\lambda^{(d')^\infty} \geq \mathbf{v} + \delta \mathbf{e}_{s'}.$$

Proof. Rewriting the hypothesis of the proposition as

$$\mathbf{r}_{d'} + \lambda \mathbf{P}_{d'}\mathbf{v} = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} + \delta \mathbf{e}_{s'} = \mathbf{v} + \delta \mathbf{e}_{s'}, \tag{5.128}$$

it follows that

$$\mathbf{r}_{d'} = (\mathbf{I} - \lambda \mathbf{P}_{d'})\mathbf{v} + \delta \mathbf{e}_{s'}.$$

Multiplying both sides of the above equality by $(\mathbf{I} - \lambda \mathbf{P}_{d'})^{-1}$ and noting that

$$(\mathbf{I} - \lambda \mathbf{P}_{d'})^{-1}\mathbf{e}_{s'} = \sum_{n=0}^{\infty} (\lambda \mathbf{P}_{d'})^n \mathbf{e}_{s'} \geq \mathbf{I}\mathbf{e}_{s'} = \mathbf{e}_{s'}$$

it follows from Theorem 5.2 that

$$\mathbf{v}_\lambda^{(d')^\infty} = (\mathbf{I} - \lambda \mathbf{P}_{d'})^{-1}\mathbf{r}_{d'} \geq \mathbf{v} + \delta \mathbf{e}_{s'}.$$

□

The above proposition guarantees that if inequality (5.127) is strict in state s', then the value of the new policy d' is strictly greater than $\mathbf{v}$ in that state. Clearly, this generalizes to improvements in multiple states.

The following important result follows immediatley from Proposition 5.9.

Theorem 5.19. In a finite state and action model, the policy iteration algorithm converges monotonically and in a finite number of iterations to an optimal policy and value function.

Proof. As a consequence of Proposition 5.9 at each iteration, either $d' = d$ or there is a strict improvement in at least one state. Since there are only finitely many stationary deterministic policies, the algorithm must terminate at some iteration with $d' = d$. In this case

$$\mathbf{v} = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} = \mathbf{r}_{d'} + \lambda \mathbf{P}_{d'}\mathbf{v}' = \underset{\delta \in D^{\mathrm{MD}}}{\text{c-max}} \{\mathbf{r}_\delta + \lambda \mathbf{P}_\delta \mathbf{v}\} = L\mathbf{v}. \tag{5.129}$$

This means that $\mathbf{v}$ is the unique solution of the Bellman equation $\mathbf{v} = L\mathbf{v}$ and d^∞ is an optimal policy since its value satisfies the Bellman equation. □

Some comments follow:

1. It follows from the above proof that an asynchronous version of policy iteration, which carries out an evaluation step as soon as a strict improvement in one state is identified, converges.

2. Theorem 5.19 provides a constructive proof of the existence of an optimal policy and a solution of the optimality equation without appealing to the Banach fixed point theorem.

3. The proof above strongly depends on the finiteness of the set of stationary deterministic policies. The next section provides a general approach for analyzing policy iteration based on relating it to Newton's method.

5.7.3 Policy iteration and Newton's method*

It is now widely recognized that policy iteration may be viewed as Newton's method for finding the zero of a differentiable function. This equivalence enables proving convergence of policy iteration when the sets of states and/or actions are non-finite, and also provides some insight into its fast convergence.

In one dimension, Newton's method (Newton-Raphson iteration) seeks to find a solution x^* of $f(x) = 0$ through iterations of the form

$$x' = x - \left(\frac{df(x)}{dx}\right)^{-1} f(x).$$

When $f(x)$ is a differentiable, convex function and its derivative is Lipschitz continuous[27], Newton's method converges quadratically[28].

Newton's method can be generalized to solve $F(\mathbf{x}) = \mathbf{0}$ where $F : \mathbb{R}^n \to \mathbb{R}^n$. In this case the Newton recursion becomes

$$\mathbf{x}' = \mathbf{x} - (J(\mathbf{x}))^{-1} F(\mathbf{x}) \tag{5.130}$$

where $J(\mathbf{x})$ denotes the *Jacobian matrix*[29] of $F(\cdot)$ evaluated at $\mathbf{x}$.

To illustrate how Newton's method arises in a Markov decision process, rewrite the problem of finding a fixed point of L as that of finding a *zero* of B where

[27] A function $f(x)$ is said to be *Lipschitz continuous* if $|f(x) - f(y)| \leq K|x - y|$ for some $K > 0$ and all x and y.

[28] As noted earlier, a real-valued sequence $x_0, x_1, x_2, \ldots$ is said to *converge quadratically* to x^* if $|x_{n+1} - x^*| \leq K|x_n - x^*|^2$ for some constant $K > 0$. This means that eventually the number of decimal places of accuracy doubles when x_{n+1} replaces x_n as an approximation of x^*.

[29] The *Jacobian matrix* of a function $F(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is an $n \times n$ matrix with its (i, j) component equal to the partial derivative of the ith component of $F(\cdot)$ with respect to the jth component of $\mathbf{x}$. The argument of $J(\cdot)$ denotes at which value of $\mathbf{x}$ it is evaluated.

$$B\mathbf{v} = L\mathbf{v} - \mathbf{v} = \underset{d \in D^{\text{MD}}}{\text{c-max}}\{\mathbf{r}_d + (\lambda \mathbf{P}_d - \mathbf{I})\mathbf{v}\} \tag{5.131}$$

for $\mathbf{v} \in V$.

From Lemma 5.5, for $\mathbf{u} \in V$ and $\mathbf{v} \in V$,

$$B\mathbf{u} \geq B\mathbf{v} + (\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I})(\mathbf{u} - \mathbf{v}) \tag{5.132}$$

where $d_\mathbf{v}$ is a $\mathbf{v}$-greedy decision rule; that is $d_\mathbf{v} \in \arg\,\text{c-max}_{d \in D^{\text{MD}}}\{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}$. This inequality generalizes the following property of differentiable, convex functions mapping $\mathbb{R}^n$ into $\mathbb{R}^n$, namely

$$F(\mathbf{y}) \geq F(\mathbf{x}) + J(\mathbf{x})(\mathbf{y} - \mathbf{x}) \tag{5.133}$$

for $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$.

The following proposition provides a **closed form** representation for the iterates of the policy iteration algorithm.

Proposition 5.10. (**Newton method representation for policy iteration**) Let $\mathbf{v}$ and $\mathbf{v}'$ be successive iterates of the policy iteration algorithm. Then

$$\mathbf{v}' = \mathbf{v} - (\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I})^{-1} B\mathbf{v}, \tag{5.134}$$

where $d_\mathbf{v}$ is any $\mathbf{v}$-greedy decision rule.

Proof. Let $d_\mathbf{v}$ be selected in (5.123) in the improvement step when entering with $\mathbf{v}$ and let $\mathbf{v}'$ be obtained at the next evaluation step by solving (5.122). Then

$$\begin{aligned}
\mathbf{v}' &= (\mathbf{I} - \lambda \mathbf{P}_{d_\mathbf{v}})^{-1} \mathbf{r}_{d_\mathbf{v}} + \mathbf{v} - \mathbf{v} \\
&= (\mathbf{I} - \lambda \mathbf{P}_{d_\mathbf{v}})^{-1} (\mathbf{r}_{d_\mathbf{v}} + (\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I})\mathbf{v}) + \mathbf{v} \\
&= \mathbf{v} + (\mathbf{I} - \lambda \mathbf{P}_{d_\mathbf{v}})^{-1} B\mathbf{v} \\
&= \mathbf{v} - (\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I})^{-1} B\mathbf{v}.
\end{aligned}$$

□

Referring to (5.130), it suggests that the matrix $\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I}$ may be viewed as the Jacobian of B evaluated at $\mathbf{v}$. This is not quite precise because B is piecewise linear, so it does not have a unique "derivative" at its break points (see Figure 5.10). However using $\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I}$ corresponding to any $\mathbf{v}$-greedy decision rule can assume this role.

The following lemma provides the basis for an easy proof that policy iteration converges in general. It shows that iterates of policy iteration are non-decreasing, bounded below by the iterates of value iteration and bounded above by the optimal value function. Hence if value iteration converges monotonically to $\mathbf{v}^*_\lambda$, which is the case if $L\mathbf{v}^0 \geq \mathbf{v}^0$, the iterates of policy iteration are "squeezed" between the iterates of value iteration and the optimal value function. This last argument is formalized

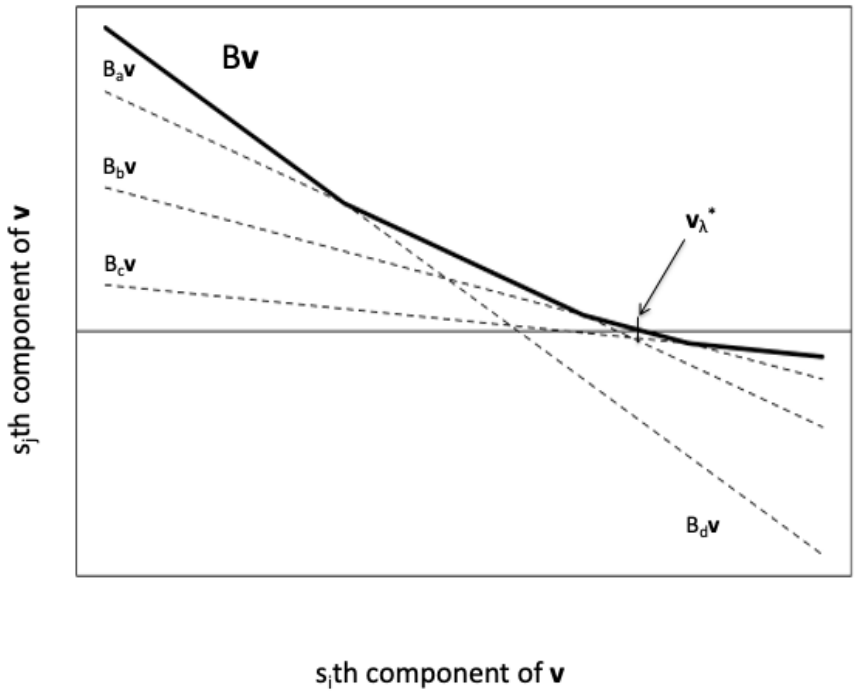


Figure 5.10: Bold line represents $B\mathbf{v}(s_j)$ as a function of $\mathbf{v}(s_i)$ for two states s_i and s_j. Observe that it is *piecewise linear and convex decreasing* with respect to $\mathbf{v}(s_i)$. Each dashed line corresponds to a component of $B_{d'}\mathbf{v}(s_j) = \mathbf{r}_{d'}(s_j) + (\lambda \mathbf{P}_{d'} - \mathbf{I})\mathbf{v}(s_j)$ as the s_i-th component of $\mathbf{v}$ varies for each of four decision rules $\{a, b, c, d\}$. The horizontal line represents $B\mathbf{v}(s_j) = 0$. The value indicated by $\mathbf{v}^*_\lambda$ corresponds to the s_ith component of the optimal value function when $B\mathbf{v}(s_j) = 0$.

in Appendix 5.13.3, which establishes the convergence of policy iteration in non-finite settings.

Lemma 5.6. For $\mathbf{v} \in V$ define $Z\mathbf{v} := \mathbf{v} - (\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I})^{-1} B\mathbf{v}$. Suppose for some $\mathbf{v} \in V$ that $B\mathbf{v} \geq \mathbf{0}$, then

1. $\mathbf{v}^*_\lambda \geq Z\mathbf{v}$,
2. $Z\mathbf{v} \geq L\mathbf{v}$,
3. $Z\mathbf{v} \geq \mathbf{v}$, and
4. $B(Z\mathbf{v}) \geq \mathbf{0}$.

Proof. The first result follows by reversing the steps in the proof of Proposition 5.10 to show that $Z\mathbf{v}$ is the value of $d^{\infty}_{\mathbf{v}}$.

The second result follows from rewriting (5.134) as

$$Z\mathbf{v} = \mathbf{v} + (\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{v}}})^{-1} B\mathbf{v} \geq \mathbf{v} + B\mathbf{v} = L\mathbf{v}$$

where the inequality follows from the result in Lemma 5.3 that for $\mathbf{u} \geq \mathbf{0}$, $(\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{v}}})^{-1}\mathbf{u} \geq \mathbf{u}$ and the assumption that $B\mathbf{v} \geq \mathbf{0}$.

The third result follows the same result in Lemma 5.3 to conclude that $(\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{v}}})^{-1} B\mathbf{v} \geq \mathbf{0}$.

To prove the fourth result, apply equation (5.132) and note that

$$Z\mathbf{v} - \mathbf{v} = (\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{v}}})^{-1} B\mathbf{v}$$

to obtain

$$B(Z\mathbf{v}) \geq B\mathbf{v} + (\lambda \mathbf{P}_{d_{\mathbf{v}}} - \mathbf{I})(Z\mathbf{v} - \mathbf{v}) = B\mathbf{v} - B\mathbf{v} = \mathbf{0}.$$

□

A proof of convergence of policy iteration in non-finite settings appears in Appendix C of this chapter.

5.8 Modified policy iteration

Modified policy iteration (MPI) algorithms provide a bridge between value iteration in which a maximization over the action set is performed in each state at each iteration and policy iteration in which a maximization takes place only after evaluating a policy by solving $\mathbf{r}_d + (\lambda \mathbf{P}_d - \mathbf{I})\mathbf{v} = \mathbf{0}$. Modified policy iteration can be viewed as either:

1. a value iteration algorithm in which the maximization is only done intermittently, or

2. a policy iteration algorithm where a policy is only evaluated approximately.

MPI is sometimes referred to as *value-oriented successive approximation* or *optimistic policy iteration*. A more descriptive alternative might be *truncated policy iteration* but this book will stick with modified policy iteration since it is widely used.

The greatest benefit of modified policy iteration with respect to value iteration occurs when there are many actions in each state, that is, when computing

$$\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}$$

is costly such as in the case of advance appointment scheduling and multi-product inventory control.

On the other hand, its main advantage with respect to policy iteration is that it avoids solving $(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{r}_d$, when the number of states is large.

5.8.1 Motivation

This exposition takes a policy iteration perspective. From Proposition 5.10, successive iterates of policy iteration may be represented by

$$\begin{aligned}\mathbf{v}' &= \mathbf{v} + (\mathbf{I} - \lambda \mathbf{P}_{d_\mathbf{v}})^{-1} B\mathbf{v} \\ &= \mathbf{v} + \sum_{n=0}^{\infty} (\lambda \mathbf{P}_{d_\mathbf{v}})^n B\mathbf{v}.\end{aligned}$$

where the second representation follows from Lemma 5.2. MPI truncates the series expansion $\sum_{n=0}^{\infty}(\lambda \mathbf{P}_{d_\mathbf{v}})^n$ at a value of m that may vary from iteration to iteration. That is, a step of MPI can be expressed as

Modified policy iteration: truncation representation

$$\mathbf{v}' = \mathbf{v} + \sum_{n=0}^{m} (\lambda \mathbf{P}_{d_\mathbf{v}})^n B\mathbf{v}. \tag{5.135}$$

Note that when $m = 0$, (5.135) becomes $\mathbf{v}' = \mathbf{v} + L\mathbf{v} - \mathbf{v} = L\mathbf{v}$, which is the value iteration recursion. And as noted above, when $m = \infty$, this is the recursive form for policy iteration. The index m may be referred to as the *truncation order* of the algorithm.

Equation (5.135) is not efficient for calculation. Instead MPI is implemented by specifying $\mathbf{v}$, performing one value iteration step $\mathbf{v}' = L\mathbf{v}$ to identify a $\mathbf{v}$-greedy decision rule, $d_\mathbf{v}$ according to

$$d_\mathbf{v} \in \underset{d \in D^{\mathrm{MD}}}{\arg\,\text{c-max}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\},$$

setting $\mathbf{u} = \mathbf{v}'$ and then applying value iteration m times using the *fixed* decision rule $d_\mathbf{v}$ as follows

$$\mathbf{u} \leftarrow L_{d_\mathbf{v}} \mathbf{u} = \mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{u}} \mathbf{u}.$$

Thus, since $L\mathbf{v} = L_{d_\mathbf{v}}\mathbf{v}$, an MPI update can be expressed as

Modified policy iteration: implementation representation

$$\mathbf{v}' = L_{d_\mathbf{v}}^m L\mathbf{v} = L_{d_\mathbf{v}}^{m+1}\mathbf{v}. \tag{5.136}$$

Note that since $d_\mathbf{v}$ is not known when implementing the above, $L\mathbf{v}$ must be evaluated to determine $d_\mathbf{v}$ and also update $\mathbf{v}$.

The next result shows that these two expressions for MPI are equivalent.

Proposition 5.11. For each $\mathbf{v} \in V$,

$$\mathbf{v} + \sum_{n=0}^{m} (\lambda \mathbf{P}_{d_\mathbf{v}})^n B\mathbf{v} = L_{d_\mathbf{v}}^m L\mathbf{v}. \tag{5.137}$$

Proof. To prove this result, expand and re-group terms as follows:

$$\begin{aligned}
\mathbf{v} + \sum_{n=0}^{m} (\lambda \mathbf{P}_{d_\mathbf{v}})^n B\mathbf{v} &= \mathbf{v} + \left(\mathbf{I} + \lambda \mathbf{P}_{d_\mathbf{v}} + (\lambda \mathbf{P}_{d_\mathbf{v}})^2 + \ldots + (\lambda \mathbf{P}_{d_\mathbf{v}})^m\right)\left(\mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}\mathbf{v} - \mathbf{v}\right) \\
&= \mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}\mathbf{v} + \lambda \mathbf{P}_{d_\mathbf{v}}(\mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}\mathbf{v} - \mathbf{v}) + \ldots + (\lambda \mathbf{P}_{d_\mathbf{v}})^m(\mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}\mathbf{v} - \mathbf{v}) \\
&= \mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}\mathbf{r}_{d_\mathbf{v}} + \ldots + (\lambda \mathbf{P}_{d_\mathbf{v}})^m \mathbf{r}_{d_\mathbf{v}} + (\lambda \mathbf{P}_{d_\mathbf{v}})^{m+1}\mathbf{v} \\
&= \mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}(\mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}(\mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}(\ldots(\mathbf{r}_{d_\mathbf{v}} + \lambda \mathbf{P}_{d_\mathbf{v}}\mathbf{v})))) \\
&= L_{d_\mathbf{v}}^{m+1}\mathbf{v} = L_{d_\mathbf{v}}^m L\mathbf{v}.
\end{aligned}$$

□

5.8.2 A modified policy iteration algorithm

This section describes an MPI algorithm using vector notation and then an implementable form using component notation. Unlike policy iteration, MPI is not a finite algorithm, therefore it requires a stopping criterion. The algorithm below uses a span-based stopping criterion that terminates the algorithm with an ϵ-optimal policy and an approximate value function that is within ϵ of the optimal value function. Figure 5.11 provides a schematic representation where m represents the order of MPI.

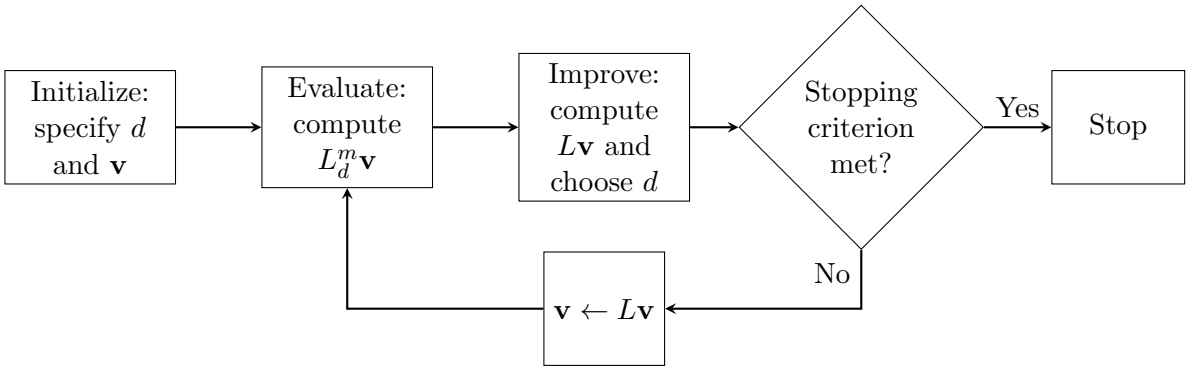


Figure 5.11: Flow diagram for modified policy iteration

The algorithm can start with either a value or a decision rule. This choice impacts whether improvement precedes evaluation or vice versa and when the stopping criterion is evaluated. The algorithm below initializes both the decision rule and value, begins with an evaluation step and checks the stopping criterion after improvement. Starting with a value only requires some modifications.

Algorithm 5.8. Modified policy iteration: vector form

1. **Initialize:**
   (a) Specify a decision rule $d' \in D^{\text{MD}}$.
   (b) Specify $\epsilon > 0$ and $\sigma > (1-\lambda)\epsilon/\lambda$.
   (c) Specify $\mathbf{v} \in V$.
   (d) Specify a sequence of non-negative integers $m_n, n = 1, 2, \ldots$.
   (e) $n \leftarrow 1$.
2. **Iterate:** While $\sigma \geq (1-\lambda)\epsilon/\lambda$:
   (a) **Update:** $d \leftarrow d'$.
   (b) **Evaluate:**
   $$\mathbf{u} \leftarrow L_d^{m_n}\mathbf{v}. \tag{5.138}$$
   (c) **Improve:** $\mathbf{v} \leftarrow L\mathbf{u}$ and choose
   $$d' \in \underset{\delta \in D^{\text{MD}}}{\arg\,\text{c-max}} \left\{\mathbf{r}_\delta + \lambda \mathbf{P}_\delta \mathbf{u}\right\} \tag{5.139}$$
   (d) $\sigma \leftarrow \text{sp}(\mathbf{v} - \mathbf{u})$.
   (e) $n \leftarrow n + 1$.
3. **Terminate:** Return $d_\epsilon = d'$ and
   $$\mathbf{v}_\lambda^\epsilon = \mathbf{v} + \frac{\lambda}{1-\lambda}(\underline{\mathbf{v} - \mathbf{u}})\mathbf{e}.$$

Some comments on the above algorithm follow:

1. The algorithm reduces to value iteration when $m_n = 0$ for $n = 1, 2, \ldots$ and policy iteration when $m_n = \infty$ for $n = 1, 2, \ldots$.
2. Specifying both a starting value and policy in the initialization avoids calculating $L\mathbf{v}$ prior to evaluation. Alternatively, if the algorithm starts without a decision rule, iteration begins with an improvement step.
3. A convenient choice for $\mathbf{v}$ is $v(s) = (1-\lambda)^{-1} \min_{a \in A_s} r(s,a)$, which is also a lower bound on $\mathbf{v}_\lambda^*$. Choosing such an initial value guarantees monotone convergence (see below).
4. The value at iteration n corresponds to the non-stationary deterministic Markovian policy that uses d_1 for m_1 periods, d_2 for $m_2 + 1$ periods, ..., d_n for $m_n + 1$

periods and then receives a terminal reward $\mathbf{v}$ where d_n denotes the decision rule evaluated at the n-th pass through the algorithm. Note if the algorithm was initiated by setting $\mathbf{u} = \mathbf{v}$ and beginning in the improve step, d_1 would be used for $m_1 + 1$ periods.

5. This general version of the algorithm keeps track of the iteration number because the pre-specified sequence m_n of orders may vary with n.

6. The impact of m_n on the speed of convergence will be explored empirically in Section 5.11.5.

7. Instead of specifying the order in advance, one might instead specify a stopping rule for the evaluation step. In other words, stop when $\text{sp}(L_d^{k+1}\mathbf{v} - L_d^k\mathbf{v})$ achieves some pre-specified tolerance. The tolerance may vary with n; presumably less accuracy is needed initially and more accuracy is needed later.

8. Note that at termination $\mathbf{v} = L\mathbf{u}$ to ensure that Theorem 5.13 applies. This specification is consistent with value iteration.

9. Similarly to value iteration, action elimination may be incorporated in the algorithm prior to applying the stopping criteria.

10. The evaluation step can be implemented using Gauss-Seidel iteration.

11. As a result of Theorem 5.13, the algorithm terminates with a decision rule corresponding to the ϵ-optimal stationary policy d_ϵ^∞ and a value function approximation that is within ϵ of the optimal value function in all components.

An implementable version

A statement of MPI in component notation follows. It is initialized by specifying an action for each state and a value to initiate the evaluation step. For transparency, the statement denotes the iterates of the evaluation step by $u(s)$. The reason for including the iteration index n is to specify the appropriate truncation order m_n at each iteration. The algorithm terminates with an ϵ-optimal policy d_ϵ^∞ and an approximation in which all components are within ϵ of the optimal value function.

Algorithm 5.9. Modified policy iteration: component form

1. **Initialize:**
   (a) For all $s \in S$, specify $a'_s \in A_s$.
   (b) Specify $\epsilon > 0$ and $\sigma > (1 - \lambda)\epsilon/\lambda$.
   (c) Specify $v(s)$ for all $s \in S$.

(d) Specify a sequence of non-negative integers $m_n, n = 1, 2, \ldots$.

(e) $n \leftarrow 1$.

2. **Iterate:** While $\sigma \geq (1-\lambda)\epsilon/\lambda$:

(a) **Update:** $a_s \leftarrow a'_s$ for all $s \in S$.

(b) **Evaluate:**

i. $m \leftarrow 1$ and $u(s) \leftarrow v(s)$ for all $s \in S$.

ii. While $m \leq m_n$:

A. For all $s \in S$,

$$u(s) \leftarrow r(s, a_s) + \lambda \sum_{j \in S} p(j|s, a_s) u(j).$$

B. $m \leftarrow m + 1$.

(c) **Improve:**

i. For all $s \in S$,

$$v(s) \leftarrow \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) u(j) \right\}$$

and choose

$$a'_s \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) u(j) \right\}.$$

ii. $\sigma \leftarrow \mathrm{sp}(\mathbf{v} - \mathbf{u})$.

iii. $n \leftarrow n + 1$.

(d) **Terminate:** For all $s \in S$, return $d_\epsilon(s) = a'_s$ and

$$v^\epsilon_\lambda(s) = v(s) + \frac{\lambda}{1-\lambda}(\underline{\mathbf{v} - \mathbf{u}}).$$

The following calculations illustrate this algorithm applied the two-state model.

Example 5.15. This examples finds an ϵ-optimal policy in the two-state model using modified policy iteration with $v(s) = 0$, $\lambda = 0.9$ and $\epsilon = 0.000001$. For simplicity, let $m_n = 3$ for all $n = 1, 2, \ldots$.

1. Choose $a'_{s_1} = a_{1,2}$ and $a'_{s_2} = a_{2,1}$ and
2. Three iterations of value iteration with the above fixed policy, yields $u(s_1) = -3.55$ and $u(s_2) = -13.55$.
3. To implement the improve step evaluate

$$\begin{aligned} v(s_1) &= \max_{i=1,2} \left\{ r(s_1, a_{1,i}) + \lambda \sum_{j=1,2} p(s_j|s_1, a_{1,i}) u(s_j) \right\} \\ &= \max\{3 + 0.9(0.8u(s_1) + 0.2u(s_2)), 5 + 0.9u(s_2)\} \\ &= \max\{-1.995, -7.195\} = -1.955, \end{aligned}$$

$$\begin{aligned} v(s_2) &= \max_{i=1,2} \left\{ r(s_2, a_{2,i}) + \lambda \sum_{j=1,2} p(s_j|s_2, a_{2,i}) u(s_j) \right\} \\ &= \max\{-5 + 0.9v(s_2), 2 + 0.9(0.4u(s_1) + 0.6u(s_2))\} \\ &= \max\{-17.195, -6.595\} = -6.595, \end{aligned}$$

and choose

$$\begin{aligned} a'_{s_1} &\in \arg\max_{i=1,2} \left\{ r(s_1, a_{1,i}) + \lambda \sum_{j=1,2} p(s_j|s_1, a_{1,i}) u(s_j) \right\} \\ &= \arg\max\{-1.995, -7.195\} = a_{1,1}, \end{aligned}$$

and

$$\begin{aligned} a'_{s_2} &\in \arg\max_{i=1,2} \left\{ r(s_2, a_{2,i} + \lambda \sum_{j=1,2} p(s_j|s_2, a_{2,i}) v(s_j) \right\} \\ &\in \arg\max\{-17.195, -6.595\} = a_{2,2}. \end{aligned}$$

4. Since $\text{sp}(\mathbf{v} - \mathbf{u}) = 5.4 \geq \lambda(1-\lambda)^{-1}\epsilon = 0.000009$, $\mathbf{v} \leftarrow \mathbf{v}'$ and return to the evaluation step.
5. Table 5.1 describes the results of four additional iterations of modified policy iteration.
6. Since $\text{sp}(\mathbf{v} - \mathbf{u}) = 0.0000032 < \lambda(1-\lambda)^{-1}\epsilon = 0.000009$, the algorithm terminates with $d_\epsilon(s_1) = a_{1,2}$ and $d_\epsilon(s_2) = a_{2,2}$.
7. The value function approximation is $v^\epsilon_\lambda(s_1) = 30.147$ and $v^\epsilon_\lambda(s_2) = 27.941$, which equal $\mathbf{v}^*_\lambda$.

Observe that it requires 5 maximizations and 15 iterations with a fixed policy to determine an ϵ-optimal policy. Note further that the ϵ-optimal policy is optimal and was first identified at iteration 2.

Iteration	$u(s_1)$	$u(s_2)$	$v(s_1)$	$v(s_2)$	a_{s_1}	a_{s_2}	$\text{sp}(\mathbf{v}' - \mathbf{v})$
2	5.2225	3.5184	8.1665	5.7800	$a_{1,2}$	$a_{2,2}$	0.6822
3	14.0232	11.8257	15.6432	13.4342	$a_{1,2}$	$a_{2,2}$	0.0115
4	19.5720	17.3663	20.6296	18.4237	$a_{1,2}$	$a_{2,2}$	0.00019
5	23.2089	21.0029	23.9027	21.6967	$a_{1,2}$	$a_{2,2}$	0.0000032

Table 5.1: Iterates of modified policy iteration in Example 5.15.

5.8.3 Convergence of modified policy iteration*

Figure 5.12 illustrates how the iterates of modified policy iteration evolve and motivates the convergence proof below. The following discussion suppresses the component index for clarity, that is, $\mathbf{u}^k$ substitutes for $\mathbf{u}^k(s_i)$ and $B\mathbf{u}^k$ for $B\mathbf{u}^k(s_j)$. Observe that:

1. Iteration starts at $\mathbf{u}^0$.

2. $\mathbf{u}^1 = \mathbf{u}^0 + B\mathbf{u}^0 = L\mathbf{u}^0$ and $\mathbf{u}^2 = \mathbf{u}^1 + B\mathbf{u}^1 = L\mathbf{u}^1$ may be regarded as the result of applying two iterates of value iteration or equivalently modified policy iteration of order 0. Since $L\mathbf{u} = B\mathbf{u} + \mathbf{u}$ the iterates of value iteration may be represented by adding the increment obtained when the "45°" line in the figure crosses the axis at 0 to the original value.

3. Iterates $\mathbf{u}^1$, $\mathbf{u}^2$ and $\mathbf{u}^3$ may be regarded as the results of applying one step of modified policy iteration or order 2. That is, the first iteration identifies the $\mathbf{u}^0$-greedy decision rule d and sets $\mathbf{u}^1 = L\mathbf{u}_0 = B\mathbf{u}^0 + \mathbf{u}^0$. The next two steps compute $\mathbf{u}^2 = L_d\mathbf{u}^1$ and $\mathbf{u}^2 = L_d\mathbf{u}^2$. We emphasize that the update is based on L_d and not L. Although they are equivalent at $\mathbf{u}^1$, using L_d instead of L avoids an extra maximization. Then the next full step of modified policy iteration of order 2 begins at $\mathbf{u}^3$ by identifying a $\mathbf{u}^3$-greedy decision rule and computing $\mathbf{u}^4 = L\mathbf{u}_3$ and then uses this decision rule to evaluate $\mathbf{u}^5$. This approach avoids two maximizations at each full MPI iteration.

The following shows that modified policy iteration converges using the same approach that was suggested for establishing the convergence of policy iteration in a non-finite setting in Appendix 5.13.3. To do so, define the operator $W_m : V \to V$ to represent the right hand side of (5.135), that is, for any integer $m \geq 0$,

$$W_m\mathbf{v} := \mathbf{v} + \sum_{n=0}^{m} (\lambda \mathbf{P}_{d_\mathbf{v}})^n B\mathbf{v}.$$

Hence the iterates of MPI can be represented by $\mathbf{v}^{n+1} = W_{m_n}\mathbf{v}^n$.

The proof of Lemma 5.6 can be easily modified to establish the analogous result for W_m. Only statement 4 requires a revised argument.

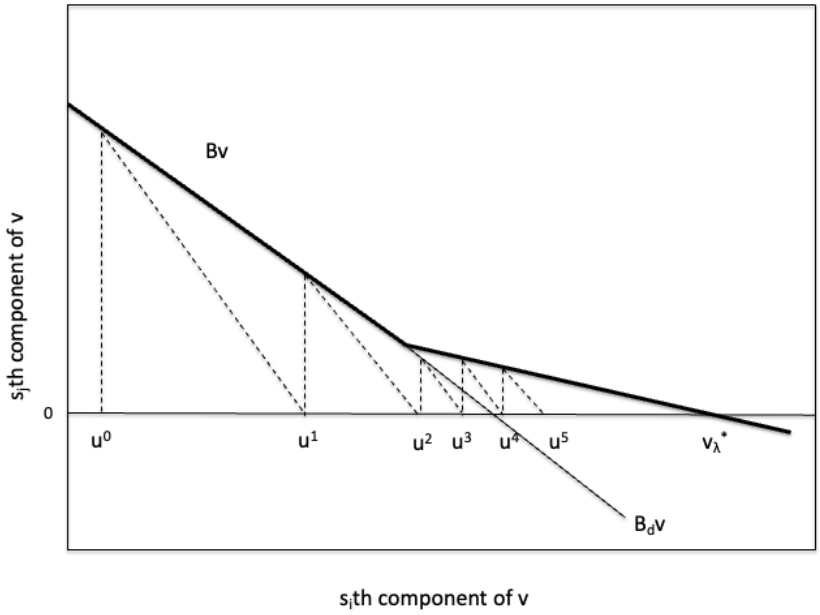


Figure 5.12: Symbolic representation of modified policy iteration updates. The bold line represents $B\mathbf{v}(s_j)$ as a function of $\mathbf{v}(s_i)$ for two states s_i and s_j and the lighter line represents $B_d\mathbf{v}(s_j)$ as a function of $\mathbf{v}(s_i)$ for the $\mathbf{u}^0$-greedy decision rule d. Note that $\mathbf{u}^2 = L_d\mathbf{u}^1 = L\mathbf{u}^2$ and $\mathbf{u}^3 = L_d\mathbf{u}^2 \neq L\mathbf{u}^2$.

Lemma 5.7. Suppose for some $\mathbf{v} \in V$ that $B\mathbf{v} \geq \mathbf{0}$ then for any integer $m \geq 0$,

1. $\mathbf{v}_\lambda^* \geq W_m\mathbf{v}$,
2. $W_m\mathbf{v} \geq L\mathbf{v}$,
3. $W_m\mathbf{v} \geq \mathbf{v}$, and
4. $B(W_m\mathbf{v}) \geq \mathbf{0}$.

The following theorem establishes the convergence of MPI using monotonicity arguments under the condition that $B\mathbf{v}^0 \geq \mathbf{0}$ or equivalently $L\mathbf{v}^0 \geq \mathbf{v}^0$. A more general result is referenced below.

Theorem 5.20. Let $m_n, n = 0, 1, \ldots$ denote a sequence of non-negative integers and let $\mathbf{v}^n, n = 0, 1, \ldots$ denote a sequence of iterates of modified policy iteration. Then if $B\mathbf{v}^0 \geq \mathbf{0}$, $\mathbf{v}^n$ converges monotonically and in norm to $\mathbf{v}^*_\lambda$.

Proof. Suppose $\mathbf{u}^0 = \mathbf{v}^0$. For each $n \geq 0$ define $\mathbf{u}^{n+1} = L\mathbf{u}^n$ and $\mathbf{v}^{n+1} = W_{m_n}\mathbf{v}^n$. Note that the sequence $\mathbf{u}^n$ corresponds to iterates of value iteration and the sequence $\mathbf{v}^n$ corresponds to iterates of MPI.

Now apply induction to establish that for all n, $B\mathbf{v}^n \geq \mathbf{0}$ and

$$\mathbf{u}^n \leq \mathbf{v}^n \leq \mathbf{v}^*_\lambda. \tag{5.140}$$

Since $B\mathbf{v} \geq \mathbf{0}$ is equivalent to $L\mathbf{v} \geq \mathbf{v}$, $L^2\mathbf{v}^0 \geq L\mathbf{v}^0 \geq \mathbf{v}^0$ so that for any n, $L^n\mathbf{v}^0 \geq \mathbf{v}^0$. That $\mathbf{v}^0 \leq \mathbf{v}^*_\lambda$ follows by letting $n \to \infty$ and noting that $L^n\mathbf{v}^0 \to \mathbf{v}^*_\lambda$. Consequently $\mathbf{v}^0 \leq \mathbf{v}^*_\lambda$.

For the induction step, assume now that (5.140) and $B\mathbf{v}^n \geq \mathbf{0}$ hold for all $m \leq n$. By the second result in Lemma 5.7 and since $L\mathbf{u}^n \leq L\mathbf{v}^n$, $\mathbf{v}^{n+1} = W_{m_n}\mathbf{v}^n \geq L\mathbf{v}^n \geq L\mathbf{u}^n = \mathbf{u}^{n+1}$. That $\mathbf{v}^{n+1} \leq \mathbf{v}^*_\lambda$ follows from the first result of Lemma 5.7 and that $B\mathbf{v}^{n+1} \geq \mathbf{0}$ follows from the fourth result of Lemma 5.7. That $\mathbf{v}^{n+1} \geq \mathbf{v}^n$ is an application of the third result of Lemma 5.7. Hence the induction hypothesis is satisfied.

Thus from (5.140) the convergence of $\mathbf{u}^n$ to $\mathbf{v}^*_\lambda$ implies the convergence of $\mathbf{v}^n$ to the same value, completing the proof. □

The proof above establishes that the iterates of value iteration (modified policy iteration of order 0) are dominated by those of modified policy iteration of order m, if $B\mathbf{v}^0 \geq \mathbf{0}$. Consequently one might conjecture that the iterates of MPI of order $m + k$ ($k > 0$) dominate those of MPI of order m. It turns out that this conjecture is false[30].

The following more general convergence result was established by Rothblum in an obscure reference in 1979 and republished as Canbolat and Rothblum [2013].

Theorem 5.21. For any $\mathbf{v}^0$, modified policy iteration converges to $\mathbf{v}^*_\lambda$.

5.9 Linear programming

Linear programming provides an elegant approach for formulating, analyzing and solving infinite horizon discounted MDPs. Moreover, it provides an effective computational approach for solving approximate dynamic programs (Chapter 9). This section describes the relationship between linear programs (LPs) and Markov decision processes and shows that key MDP concepts have direct analogs in linear programming. The main result is that an optimal solution to a (dual) linear program corresponds to an

[30] See example 6.5.1 in Puterman [1994].

optimal policy to the Markov decision process. This long section requires background in linear algebra and linear programming, some of which appears in Appendix C.

5.9.1 The primal linear program

If $v(s)$ satisfies the Bellman equation

$$v(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}$$

for all $s \in S$, then it satisfies the system of inequalities

$$v(s) \geq \; r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \tag{5.141}$$

for all $a \in A_s$ and $s \in S$.

By Corollary 5.2, any $v(s)$ satisfying (5.141) for all $a \in A_s$ and $s \in S$ is an upper bound on $v^*_\lambda(s)$. Hence a solution of the Bellman equation may be thought of as the *smallest* (in a component-wise sense) $v(s)$ that satisfies (5.141) for all $a \in A_s$ and $s \in S$. To formulate the problem of finding such a $v(s)$ using linear programming requires a single objective function. To obtain one, selecting $\alpha(s) > 0$ for all $s \in S$ leads to the following linear program.

Primal LP formulation of a discounted MDP

$$\text{minimize} \quad \sum_{s \in S} \alpha(s) v(s) \tag{5.142a}$$

$$\text{subject to} \quad v(s) - \lambda \sum_{j \in S} p(j|s,a) v(j) \geq r(s,a), \quad a \in A_s, s \in S \tag{5.142b}$$

Some comments about this formulation follow:

1. The above model is often referred to as the *primal* linear program. Associated with every primal linear is a *dual* linear program, which is described below. Note that the relationship between policies and values becomes more apparent through analysis of the dual linear program.

2. In this formulation, $v(s)$ is *unrestricted* in sign for all $s \in S$. That is, it is not restricted to be non-negative as is often the case in LP formulations.

3. This LP has $|S|$ variables and $\sum_{s \in S} |A_s|$ constraints. Thus, the number of constraints might far exceed the number of variables.

4. Choosing $\alpha(s)$ so that $\sum_{s\in S}\alpha(s) = 1$ allows interpretation of this quantity as a probability distribution on S.

5. Note that there is exactly one positive coefficient in each row[31] corresponding to the coefficient of $v(s)$ with value $1 - \lambda p(s|s, a)$.

6. Since $v^*_\lambda(s)$ for all $s \in S$ satisfies the Bellman equation, it is a feasible[32] solution for the primal LP.

The following example illustrates this primal LP formulation.

Example 5.16. This example formulates the primal LP for the two-state example. The Bellman equations are given in equations (5.62) and (5.63). The primal linear program is given by

$$\begin{aligned} [1]\text{minimize} \quad & \alpha(s_1)v(s_1) + \alpha(s_2)v(s_2) \\ \text{subject to} \quad & (1 - 0.8\lambda)v(s_1) - 0.2\lambda v(s_2) \geq 3 \\ & 1v(s_1) - \lambda v(s_2) \geq 5 \\ & 0v(s_1) + (1 - \lambda)v(s_2) \geq -5 \\ & (1 - 0.4\lambda)v(s_1) - 0.6\lambda v(s_2) \geq 2. \end{aligned} \tag{5.143}$$

The four constraints correspond respectively to choosing actions $a_{1,1}$ or $a_{1,2}$ in s_1 and $a_{2,1}$ or $a_{2,2}$ in s_2. For $\lambda = 0.9$, the four constraints can be written as:

$$\begin{aligned} v(s_2) &\leq \frac{14}{9}v(s_1) - \frac{50}{3} \\ v(s_2) &\leq \frac{10}{9}v(s_1) - \frac{50}{9} \\ v(s_2) &\geq -50 \\ v(s_2) &\geq \frac{18}{23}v(s_1) + \frac{100}{23}. \end{aligned}$$

Written as above, it is easy to visualize the feasible region in two dimensions. The first and fourth constraints intersect at $(v(s_1), v(s_2)) = (27.1875, 25.625)$, while the second and fourth constraints intersect at $(v(s_1), v(s_2)) = (30.147, 27.941)$. Since the objective is to maximize the value function in each state, the optimal solution is the latter, which is the same value function found by value iteration in Example 5.11. The figure below illustrates the region described by the above constraints together with a graphical solution for $\lambda = 0.9$.

Note that the optimal policy can be obtained by looking at which constraints are tight (hold with equality) at the optimal solution. The second constraint cor-

[31] A linear system with this property is referred to as *Leontief substitution system*.

[32] A feasible solution is one that satisfies all the LP constraints.

responds to choosing $a_{1,2}$ in s_1 and the fourth constraint corresponds to choosing $a_{2,2}$ in s_2, again matching the policy identified from previous methods.

Finally, note that the optimal solution corresponds to the "lower left" vertex of the feasible region. So any values of $\alpha(s_1) > 0$ and $\alpha(s_2) > 0$ will result in the same optimal solution, as will be show formally in Corollary 5.5.

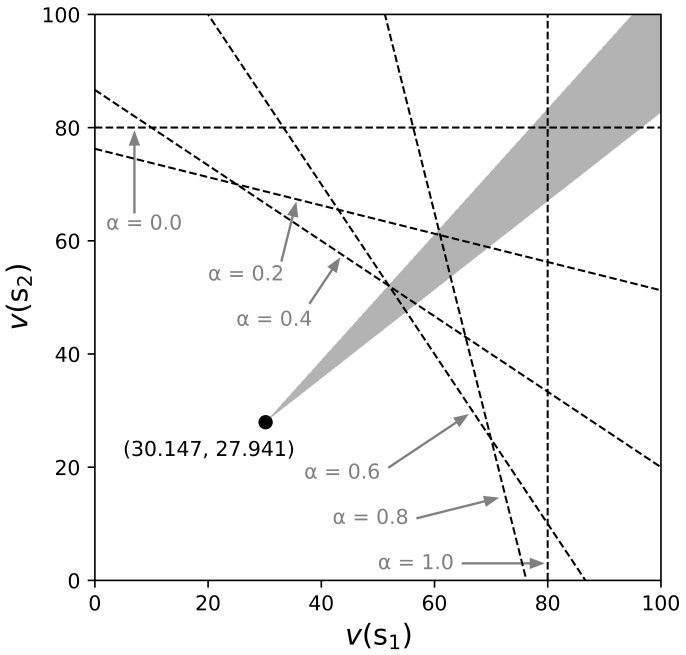


Figure 5.13: Figure for Example 5.16. Shaded region represents the feasible region, defined by the constraints. This figure assumes that $\alpha(s_1) = \alpha$ and $\alpha(s_2) = 1 - \alpha$ for some $\alpha > 0$. Different α values represent different objective functions. The identified vertex is the optimal solution for all α.

LP formulation: vector notation*

This section discusses the matrix-vector formulation of the primal linear program. Unfortunately, the Bellman equation expressed in terms of "c-max" as

$$\mathbf{v} = \underset{d \in D^{\text{MD}}}{\text{c-max}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}, \tag{5.144}$$

does not have a direct analog in terms of vector inequalities. If instead, one considered the Bellman equation expressed as

$$\mathbf{v} = \max_{d \in D^{\text{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}, \tag{5.145}$$

where the maximum was taken over all Markov deterministic decision rules, then this would have a direct analog as

$$\mathbf{v} \geq \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} \tag{5.146}$$

for all $d \in D^{\text{MD}}$. Choosing this as a starting point and letting $\boldsymbol{\alpha} > \mathbf{0}$ denote a vector of length $|S|$ leads to the following linear program.

Primal LP formulation of a discounted MDP: vector notation

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\alpha}^{\mathsf{T}} \mathbf{v} && \text{(5.147a)} \\ \text{subject to} \quad & (\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} \geq \mathbf{r}_d, \quad d \in D^{\text{MD}}. && \text{(5.147b)} \end{aligned}$$

While elegant, this formulation is impractical for computation because it requires enumeration of all decision rules, which runs contrary to Markov decision process principles. Moreover it repeats constraints when two or more decision rules choose the same action in a particular state. For example, consider the decision rules $d_1 = \{a_{1,1}, a_{2,1}\}$ and $d_2 = \{a_{1,1}, a_{2,2}\}$ in the two-state model. Since both choose the same action in state s_1, the inequalities $\mathbf{v} \geq \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \mathbf{v}$ and $\mathbf{v} \geq \mathbf{r}_{d_2} + \lambda \mathbf{P}_{d_2} \mathbf{v}$ would both contain the same constraint involving $a_{1,1}$ in s_1.

Existence of a solution of the primal linear program*

The following result shows that the primal LP has an optimal solution, and that solving the LP finds the optimal value function $\mathbf{v}^*_\lambda$ and an optimal policy for the MDP. Its proof is subtle and uses the structural result that the primal LP formulation of an MDP has exactly one positive coefficient in each row.

Theorem 5.22. Let $\boldsymbol{\alpha} > \mathbf{0}$ and assume $0 \leq \lambda < 1$.

1. The linear program (5.142) has an optimal solution $v^*(s)$ for all $s \in S$.
2. For each $s \in S$ there exists an $a^*_s \in A_s$ for which
$$v^*(s) - \lambda \sum_{j \in S} p(j|s, a^*_s) v^*(j) = r(s, a^*_s). \tag{5.148}$$
3. For each $s \in S$, let $d^*(s) = a^*_s$ as defined in (5.148). Then $(d^*)^\infty$ is an optimal policy.
4. The solution $v^*(s) = v^*_\lambda(s)$ for all $s \in S$ is the unique optimal solution of the LP.

Proof. Define $R = \max_{a \in A_s, s \in S} r(s,a)$ and let $v'(s) = (1-\lambda)^{-1}R$ for all $s \in S$. Then direct substitution into (5.142b) gives

$$v'(s) - \lambda \sum_{j \in S} p(j|s,a)v'(j) = R \geq r(s,a)$$

for all $a \in A_s, s \in S$. Hence, the linear program is feasible ($\mathbf{v}'$ is a feasible solution). Now, consider any feasible solution $\mathbf{v}$. For every $d \in D^{\text{MD}}$, it satisfies $(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} \geq \mathbf{r}_d$ and therefore $\mathbf{v} \geq (\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{r}_d$, by the second result of Lemma 5.3. Since $\boldsymbol{\alpha} > \mathbf{0}$, $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{v} \geq \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{r}_d$ for all $d \in D^{\text{MD}}$. In other words, the objective function value of every feasible $\mathbf{v}$ is bounded below by $\max_{d \in D^{\text{MD}}} \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{r}_d$. Hence by Theorem C.1 in Appendix C, the linear program has an optimal solution, denoted by $v^*(s)$ for all $s \in S$.

To prove the second result, assume to the contrary that there exists an $s' \in S$ such that for all $a \in A_{s'}$

$$v^*(s') - \lambda \sum_{j \in S} p(j|s',a)v^*(j) > r(s',a). \tag{5.149}$$

Hence there exists an $\epsilon > 0$ for which

$$v^*(s') - \lambda \sum_{j \in S} p(j|s',a)v^*(j) - \epsilon \geq r(s',a).$$

Since $\lambda p(s'|s',a)\epsilon > 0$, it follows that

$$(v^*(s') - \epsilon) - \lambda \left(\sum_{j \in S \setminus \{s'\}} p(j|s',a)v^*(j) + p(s'|s',a)(v^*(s') - \epsilon) \right) \geq v^*(s') - \lambda \sum_{j \in S} p(j|s',a)v^*(j) - \epsilon$$

Hence, the value function defined by

$$v'(s) := \begin{cases} v^*(s) & s \neq s' \\ v^*(s) - \epsilon & s = s' \end{cases}$$

is a feasible solution for the linear program. Since $\alpha(s) > 0$ for all $s \in S$, this implies the value function $v'(s)$ has a lower objective function value than that of $v^*(s)$ (by $\alpha(s')\epsilon$), which contradicts the optimality of $v^*(s)$. Hence, for each $s \in S$ there exists an $a_s^* \in A_s$ for which

$$v^*(s) - \lambda \sum_{j \in S} p(j|s,a_s^*)v^*(j) = r(s,a_s^*).$$

To prove the third result, the second result and Theorem 5.2 imply that $v^*(s) = v_\lambda^{(d^*)^\infty}(s)$ where $d^*(s) = a_s^*$ for all $s \in S$. By Theorem 5.7, any feasible solution of the

LP must be an upper bound on $v^*_\lambda(s)$ for all $s \in S$. In particular, $v_\lambda^{(d^*)^\infty}(s) \geq v^*_\lambda(s)$ for all $s \in S$, which implies that the stationary policy on $d^*(s)$ is an optimal policy. Note also that $v^*_\lambda(s) \geq v_\lambda^{(d^*)^\infty}(s)$ for all $s \in S$ by definition of $v^*_\lambda(s)$. Hence, $v^*(s) = v_\lambda^{(d^*)^\infty}(s) = v^*_\lambda(s)$. Since $\mathbf{v}^*_\lambda$ is the unique solution to the Bellman equation, it is also the unique solution to the primal LP, proving the fourth result. □

The above theorem establishes the existence of a solution to the Bellman equation without using the Banach fixed point theorem (Theorem 5.5) and as well the existence of a stationary optimal policy. Moreover it establishes that for each $s \in S$ there exists at least one action for which the inequality

$$v^*(s) - \lambda \sum_{j \in S} p(j|s,a) v^*(j) \geq r(s,a) \tag{5.150}$$

holds with equality. Hence solving the LP and noting which constraints hold with equality identifies an optimal policy. When two or more actions satisfy (5.150) in some state $s \in S$, any stationary policy that randomizes between them in state s is optimal.

It may appear that the optimal solution and policy depend on the particular choice of $\boldsymbol{\alpha}$. However, the proof of Theorem 5.22 remains valid for any $\boldsymbol{\alpha} > \mathbf{0}$, yielding the following important result.

Corollary 5.5. Fix $\boldsymbol{\alpha} > \mathbf{0}$. Let $\mathbf{v}^*$ be an optimal solution to the LP and $(d^*)^\infty$ be a corresponding optimal policy. Then $\mathbf{v}^*$ and $(d^*)^\infty$ are optimal for any other $\boldsymbol{\alpha} > \mathbf{0}$.

This result suggests that the solution of the primal LP lies at the "lower left most" vertex of the feasible region defined by the constraints (5.142b). Furthermore, the feasible region must be contained in the region[33]

$$\{\mathbf{v} \in V \mid \mathbf{v} = \mathbf{v}^* + \mathbf{y}, \mathbf{y} \geq \mathbf{0}\}.$$

5.9.2 The dual linear program

Analyzing a Markov decision process through its dual linear program provides a direct and elegant way of identifying optimal policies and insight into the relationship between between feasible solutions and stationary policies. The key result of this section is Theorem 5.27, which shows that there is a one-to-one correspondence between optimal solutions to the dual LP and optimal policies of the MDP.

Appealing to LP duality theory (see Appendix C), the dual linear program for the discounted MDP may be expressed as:

[33] This region is a *convex cone*, a fundamental concept in linear programming.

Dual LP formulation of discounted MDP

$$\text{maximize} \quad \sum_{s\in S}\sum_{a\in A_s} r(s,a)x(s,a) \tag{5.151a}$$

$$\text{subject to} \quad \sum_{a\in A_s} x(s,a) - \sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a)x(j,a) = \alpha(s), \quad s\in S \tag{5.151b}$$

$$x(s,a) \geq 0, \quad a\in A_s, s\in S \tag{5.151c}$$

Some comments regarding this formulation follow.

1. The dual formulation reverses the role of objective function and right-hand side constraint coefficients. Observe that $\alpha(s)$ appears on the right-hand side of (5.151b) as constraints and $r(s,a)$ appears as coefficients in the objective function (5.151a).

2. Note that in equation (5.151b), the index j represents the starting state and the index s represents the state reached after one transition. The reflects the observation below that a matrix representation of the dual may be expressed in terms of the transpose of the transition probability matrix.

3. In contrast to the primal, the objective function of the dual is maximized, instead of minimized.

4. The dual linear program has $|S|$ rows and $\sum_{s\in S}|A_s|$ variables. Hence, usually the number of columns far exceeds the number of rows.

5. In contrast to the primal variables, the dual variables are constrained to be non-negative.

The following example provides the dual formulation of the linear program for the two-state model.

Example 5.17. The dual linear program of (5.143) can be written as :

$$\begin{aligned}
\text{minimize} \quad & 3x(s_1,a_{1,1}) + 5x(s_1,a_{1,2}) - 5x(s_2,a_{2,1}) + 2x(s_2,a_{2,2}) \\
\text{subject to} \quad & x(s_1,a_{1,1}) + x(s_1,a_{1,2}) - 0.8\lambda x(s_1,a_{1,1}) - 0.4\lambda x(s_2,a_{2,2}) = \alpha(s_1) \\
& x(s_2,a_{2,1}) + x(s_2,a_{2,2}) - 0.2\lambda x(s_1,a_{1,1}) \\
& \quad - \lambda x(s_1,a_{1,2}) - \lambda x(s_2,a_{2,1}) - 0.6\lambda x(s_2,a_{2,2}) = \alpha(s_2) \\
& x(s_1,a_{1,1}), x(s_1,a_{1,2}), x(s_2,a_{2,1}), x(s_2,a_{2,2}) \geq 0.
\end{aligned}$$

Letting $\lambda = 0.9$ and $\alpha(s_1) = \alpha = 0.5 = 1 - \alpha(s_2)$, one can use any LP solver to find the optimal solution $x^*(s_1,a_{1,1}) = 0, x^*(s_1,a_{1,2}) = 3.015, x^*(s_2,a_{2,1}) = 0$

and $x^*(s_2, a_{2,2}) = 6.985$. The corresponding dual objective function value is $\sum_s \sum_{a \in A_s} x^*(s,a) r(s,a) = 29.044$.

Recall from Example 5.16 that $v^*(s_1) = 30.147$ and $v^*(s_2) = 27.941$. Thus,

$$\sum_{s \in S} \alpha(s) v^*(s) = 29.044 = \sum_{s \in S} \sum_{a \in A_s} x^*(s,a) r(s,a).$$

This result, where the primal and dual optimal solutions have the same objective function value, is known as the Strong Duality Theorem (see Theorem 5.26 and Theorem C.6).

Where convenient, the dual variables $x(s,a), a \in A_s, s \in S$ will be represented by the vector $\mathbf{x}$. One may think of $\mathbf{x}$ as a "long" vector, where the first $|A_{s_1}|$ components correspond to $x(s_1,a), a \in A_{s_1}$, the second $|A_{s_2}|$ components correspond to $x(s_2,a), a \in A_{s_2}$, and so on.

5.9.3 Dual feasible solutions and stationary policies

This section establishes an elegant result relating feasible solutions to the dual linear program to randomized stationary policies in a Markov decision process. To do so, for each $d \in D^{\mathrm{MR}}, a \in A_s$ and $s \in S$, define $x_d(s,a)$ by:

$$x_d(s,a) := \sum_{j \in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} P^{d^\infty}[X_n = s, Y_n = a \,|\, X_1 = j]. \tag{5.152}$$

Let $\mathbf{x}_d$ denote the corresponding vector representation for $x_d(s,a)$.

Assuming that $\sum_{j \in S} \alpha(j) = 1$, the quantity $x_d(s,a)$ can be interpreted as the discounted joint probability under decision rule d^∞, that the process is in state s and chooses action a over the infinite horizon when the process starts in state $j \in S$ with probability $\alpha(j)$.

The following lemma is fundamental when constructing a randomized decision rule from a feasible solution to the dual linear program using (5.153) below.

Lemma 5.8. Suppose $\alpha(s) > 0$ for all $s \in S$. Then if $x(s,a)$ is a feasible solution to the dual linear program

$$\sum_{a \in A_s} x(s,a) > 0.$$

Proof. From (5.151b) and the condition that $x(s,a) \geq 0$ for all $a \in A_s, s \in S$ it follows

immediately that for all $s \in S$,

$$\sum_{a \in A_s} x(s,a) \geq \alpha(s) > 0.$$

□

Given a feasible solution $x(s,a), a \in A_s, s \in S$ to the dual linear program, define the Markovian randomized decision rule $d_{\mathbf{x}}$ by:

$$w_{d_{\mathbf{x}}}(a|s) := \frac{x(s,a)}{\sum_{a' \in A_s} x(s,a')}. \tag{5.153}$$

As a consequence of Lemma 5.8, the denominator is positive so that $w_{d_{\mathbf{x}}}(a|s)$ is well defined. Recall that $w_{d_{\mathbf{x}}}(a|s)$ denotes the probability that the randomized decision rule $d_{\mathbf{x}}$ chooses action a in state s. Consequently, if for each $s \in S$, $x(s,a) > 0$ for exactly one $a \in A_s$, $d_{\mathbf{x}}$ is deterministic.

Feasible solutions and randomized stationary policies

The following theorem establishes the relationship between solutions to the dual and randomized stationary policies. More specifically, when $\mathbf{x}_{d_{\mathbf{x}}}$ is constructed by the following process:

1. obtain a feasible solution to the dual $\mathbf{x}$,
2. construct $d_{\mathbf{x}}(s)$ using (5.153),
3. compute $x_{d_{\mathbf{x}}}(s,a), a \in A_s, s \in S$ by (5.152),

it follows that $\mathbf{x}_{d_{\mathbf{x}}} = \mathbf{x}$. Part 2 of the following theorem shows that this relationship holds. Its proof is straightforward but intricate.

Theorem 5.23. Suppose $0 \leq \lambda < 1$ and $\alpha(s) > 0$ for all $s \in S$. Then

1. For any $d \in D^{\mathrm{MR}}$, if $x_d(s,a)$ is defined by (5.152) for all $a \in A_s$ and $s \in S$, then it is a feasible solution to the dual LP.
2. Suppose $x(s,a)$ is a feasible solution to the dual LP and define $d_{\mathbf{x}}$ by (5.153). Then $x_{d_{\mathbf{x}}}(s,a) = x(s,a)$ for all $a \in A_s$ and $s \in S$.

Proof. By definition $x_d(s,a) \geq 0$ for all $a \in A_s$, $s \in S$. Write the expression on the left hand side of the equality constraint (5.151b) as

$$\begin{aligned}
\sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a)x_d(j,a) &= \sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a) \sum_{k\in S}\alpha(k)\sum_{n=1}^{\infty}\lambda^{n-1}P^{d^\infty}[X_n = j, Y_n = a\,|X_1 = k] \\
&= \sum_{k\in S}\alpha(k)\sum_{n=1}^{\infty}\lambda^{n}P^{d^\infty}[X_{n+1} = s\,|X_1 = k] \\
&= \sum_{k\in S}\alpha(k)\left(\sum_{n=1}^{\infty}\lambda^{n-1}P^{d^\infty}[X_n = s\,|X_1 = k] - P^{d^\infty}[X_1 = s\,|X_1 = k]\right) \\
&= \sum_{a\in A_s}\sum_{k\in S}\alpha(k)\sum_{n=1}^{\infty}\lambda^{n-1}P^{d^\infty}[X_n = s, Y_n = a\,|X_1 = k] - \alpha(s) \\
&= \sum_{a\in A_s} x_d(s,a) - \alpha(s).
\end{aligned}$$

Rearranging terms shows that $x_d(s,a)$ is a feasible solution to the dual LP.

To prove the second result, assume $x(s,a)$ is a feasible solution to the dual LP. Let

$$u(s) = \sum_{a\in A_s} x(s,a), \tag{5.154}$$

which by Lemma 5.8 is strictly positive for each $s \in S$. From (5.151b) it follows that

$$\begin{aligned}
\alpha(s) &= u(s) - \sum_{j\in S}\sum_{a\in A_j}\lambda p(s|j,a)x(j,a) \\
&= u(s) - \sum_{j\in S}\sum_{a\in A_j}\lambda p(s|j,a)x(j,a)\frac{u(j)}{\sum_{a\in A_j}x(j,a)} \\
&= u(s) - \sum_{j\in S}\sum_{a\in A_j}\lambda p(s|j,a)w_{d_{\mathbf{x}}}(a|j)u(j) \\
&= u(s) - \sum_{j\in S}\lambda P_{d_{\mathbf{x}}}(s|j)u(j),
\end{aligned}$$

where $P_{d_{\mathbf{x}}}(s|j)$ is defined in (5.3). Writing the last expression in matrix notation yields

$$\boldsymbol{\alpha}^{\mathsf{T}} = \mathbf{u}^{\mathsf{T}}(\mathbf{I} - \lambda\mathbf{P}_{d_{\mathbf{x}}}).$$

From Lemma 5.2

$$\mathbf{u}^{\mathsf{T}} = \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{I} - \lambda\mathbf{P}_{d_{\mathbf{x}}})^{-1} = \boldsymbol{\alpha}^{\mathsf{T}}\left(\sum_{n=1}^{\infty}\lambda^{n-1}\mathbf{P}_{d_{\mathbf{x}}}^{n-1}\right). \tag{5.155}$$

Expressing (5.155) in component notation gives

$$u(s) = \sum_{j\in S}\alpha(j)\sum_{n=1}^{\infty}\lambda^{n-1}P_{d_{\mathbf{x}}}^{n-1}(s|j)$$

$$= \sum_{j \in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} \sum_{a \in A_s} P^{d^\infty_{\mathbf{x}}}[X_n = s, Y_n = a \,|X_1 = j] \tag{5.156}$$

$$= \sum_{a \in A_s} x_{d_{\mathbf{x}}}(s, a) \tag{5.157}$$

for all $s \in S$, where (5.156) is a consequence of the law of total probability applied as follows:

$$P^{n-1}_{d_{\mathbf{x}}}(s|j) = P^{d^\infty_{\mathbf{x}}}[X_n = s \,|X_1 = j] = \sum_{a \in A_s} P^{d^\infty_{\mathbf{x}}}[X_n = s, Y_n = a \,|X_1 = j].$$

This establishes that

$$\sum_{a \in A_s} x(s, a) = \sum_{a \in A_s} x_{d_{\mathbf{x}}}(s, a). \tag{5.158}$$

To complete the proof, from the definition of $x_{d_{\mathbf{x}}}(s, a)$, for all $s \in S$ and $a \in A_s$,

$$x_{d_{\mathbf{x}}}(s, a) = \sum_{j \in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} P^{d^\infty_{\mathbf{x}}}[X_n = s, Y_n = a \,|X_1 = j]$$

$$= \sum_{j \in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} P^{d^\infty_{\mathbf{x}}}[X_n = s|X_1 = j] w_{d_{\mathbf{x}}}(a|s)$$

$$= \sum_{a \in A_s} x_{d_{\mathbf{x}}}(s, a) w_{d_{\mathbf{x}}}(a|s) \tag{5.159}$$

$$= \sum_{a \in A_s} x_{d_{\mathbf{x}}}(s, a) \frac{x(s, a)}{\sum_{a' \in A_s} x(s, a')} \tag{5.160}$$

$$= x(s, a). \tag{5.161}$$

where the third equality (5.159) is a consequence of (5.157) and the last equality is due to (5.158). □

Basic feasible solutions and deterministic stationary policies

Theorem 5.23 established that any stationary randomized policy generates a feasible solution to the dual problem, and that all feasible solutions to the dual problem can be generated by stationary randomized policies.

The following result establishes a correspondence between *basic* feasible solutions[34] to the dual problem and stationary *deterministic* policies.

[34] A basic feasible solution is an extreme point of the region defined by the constraints of the linear program. This means that it cannot be expressed as a convex combination of any other points in the region.

Theorem 5.24. Suppose $0 \leq \lambda < 1$ and $\alpha(s) > 0$ for all $s \in S$. Then

1. For any $d \in D^{\text{MD}}$, if $x_d(s,a)$ is defined by (5.152) for all $a \in A_s$, and $s \in S$ then it is a basic feasible solution to the dual LP.

2. Suppose $x(s,a)$ is a basic feasible solution to the dual LP. Then $d_{\mathbf{x}} \in D^{\text{MD}}$ and $d_{\mathbf{x}}(s) = a_s$ if $x(s,a_s) > 0$.

Proof. If $d \in D^{\text{MD}}$, it follows from Theorem 5.23 that $x_d(s,a)$ is feasible to the dual linear program.

Suppose to the contrary that $x_d(s,a)$ is not a basic feasible solution. Then there exists basic feasible solutions $y(s,a)$ and $z(s,a)$, with $y(s,a) \neq z(s,a)$ for some $a \in A_s$ and $s \in S$ and $\beta \in (0,1)$ for which

$$x_d(s,a) = \beta y(s,a) + (1-\beta) z(s,a)$$

for all $a \in A_s$ and $s \in S$. From Lemma 5.8,

$$\sum_{a \in A_s} y(s,a) > 0 \text{ and } \sum_{a \in A_s} z(s,a) > 0$$

for each s. And since they are distinct solutions, there must be some $\hat{s}$ such that $y(\hat{s},a') > 0$ and $z(\hat{s},a'') > 0$ where a' and a'' are distinct actions in $A_{\hat{s}}$. Since $x(s,a)$ is a convex combination of $y(s,a)$ and $z(s,a)$, $x(\hat{s},a)$ must be positive for at least two distinct actions, implying that $d_{\mathbf{x}}$ is a randomized policy, which is a contradiction.

To prove the second result, first note that for each s, since $\sum_{a \in A_s} x(s,a) > 0$, $x(s,a) > 0$ for at least one $a \in A_s$. If $x(s,a)$ is a basic feasible solution, then it has at most $|S|$ positive components, since the dual problem has $|S|$ equality constraints. Since $\alpha(s) > 0$ for all $s \in S$, $x(s,a) > 0$ for exactly one $a \in A_s$. Thus for each $s \in S$, $d_{\mathbf{x}}(s)$ defined by (5.153) chooses one action in A_s with probability 1, so that it is deterministic. □

Figure 5.14 provides further insight into the consequences of Theorems 5.23 and 5.24. The shaded polyhedron[35] represents the set of feasible solutions to the dual LP. Vertices of the polyhedron are basic feasible solutions, which correspond to stationary deterministic policies. All other points in the polyhedron are feasible solutions, which correspond to stationary randomized policies. Any point in the polyhedron can be written as a convex combination of the vertices, which is equivalent to the fact that randomized policies may be viewed as weighted combinations of deterministic policies.

To make things concrete, consider the two vertices $\mathbf{x}^1$ and $\mathbf{x}^2$ shown in Figure 5.14. They correspond to Markovian deterministic decision rules d_1 and d_2 that choose actions $d_1(s)$ and $d_2(s)$, respectively, in state s. Now, consider a feasible solution

[35] A *polyhedron* is a region formed by the intersection of a finite number of linear inequalities. See Appendix C for definitions of additional linear programming concepts.

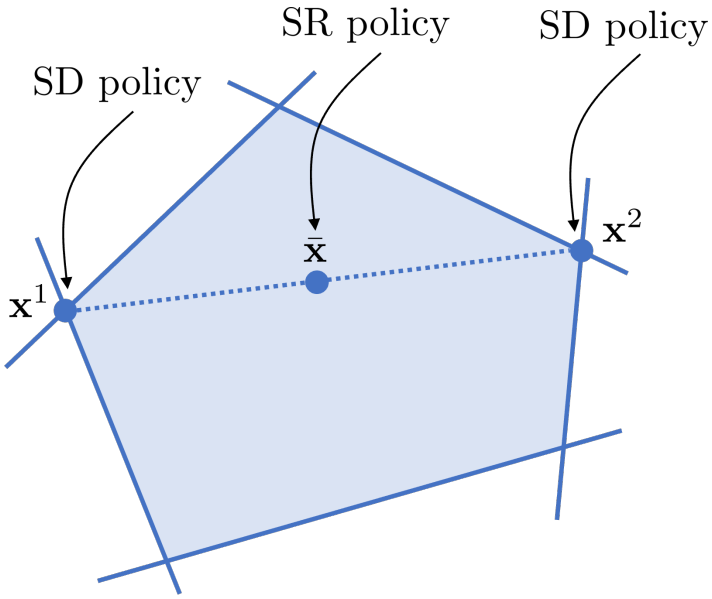


Figure 5.14: Relationship between feasible solutions of the dual LP and stationary policies of the MDP. Basic feasible solutions $\mathbf{x}^1$ and $\mathbf{x}^2$ correspond to stationary deterministic policies. Their convex combination, $\bar{\mathbf{x}}$, corresponds to a stationary randomized policy.

$\bar{\mathbf{x}} = \mu\mathbf{x}^1 + (1-\mu)\mathbf{x}^2$, where $\mu \in (0,1)$, with corresponding Markovian randomized decision rule $d_{\bar{\mathbf{x}}}$. From equation (5.153),

$$w_{d_{\bar{\mathbf{x}}}}(a|s) = \frac{\bar{x}(s,a)}{\sum_{a' \in A_s} \bar{x}(s,a')} \tag{5.162}$$

$$= \frac{\mu x^1(s,a) + (1-\mu)x^2(s,a)}{\sum_{a' \in A_s} (\mu x^1(s,a') + (1-\mu)x^2(s,a'))} \tag{5.163}$$

By definition, for $i = 1, 2$

$$x^i(s,a) = \begin{cases} \sum_{j \in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} P^{d_i^\infty}[X_n = s \,|X_1 = j] & \text{if } a = d_i(s) \\ 0 & \text{if } a \neq d_i(s). \end{cases} \tag{5.164}$$

Hence

$$w_{d_{\bar{\mathbf{x}}}}(a|s) = \begin{cases} \frac{\mu x^1(s,d_1(s))}{\mu x^1(s,d_1(s)) + (1-\mu)x^2(s,d_2(s))} & \text{if } a = d_1(s) \\ \frac{(1-\mu)x^2(s,d_2(s))}{\mu x^1(s,d_1(s)) + (1-\mu)x^2(s,d_2(s))} & \text{if } a = d_2(s) \\ 0 & \text{otherwise.} \end{cases} \tag{5.165}$$

Given this 1-1 relationship between feasible solutions of the dual LP and policies of the MDP, and the fact that randomized policies are weighted averages of deterministic policies, it follows that the dual feasible region must be bounded. The proof is left as

an exercise. This result allows us to provide a first principles proof of the existence of solutions to the dual LP.

Theorem 5.25. The feasible region of the dual LP is bounded.

As a preview of the results in the next section, the following result establishes a connection between primal solutions (value functions) and dual solutions (discounted joint probability distributions).

Proposition 5.12. Let $\mathbf{x}$ be a dual feasible solution and $d_{\mathbf{x}} \in D^{\text{MR}}$ be the randomized policy defined by (5.153). Then,

$$\sum_{s\in S}\sum_{a\in A_s} x_{d_{\mathbf{x}}}(s,a)r(s,a) = \sum_{j\in S}\alpha(j)v_{\lambda}^{d_{\mathbf{x}}^{\infty}}(j). \tag{5.166}$$

Proof. From equations (5.14) and (5.152),

$$v_{\lambda}^{d_{\mathbf{x}}^{\infty}}(j) = \sum_{s\in S}\sum_{a\in A_s}\sum_{n=1}^{\infty}\lambda^{n-1}P^{d_{\mathbf{x}}^{\infty}}[X_n = s, Y_n = a\,|X_1 = j]r(s,a), \tag{5.167}$$

and

$$x_{d_{\mathbf{x}}}(s,a) = \sum_{j\in S}\alpha(j)\sum_{n=1}^{\infty}\lambda^{n-1}P^{d_{\mathbf{x}}^{\infty}}[X_n = s, Y_n = a\,|X_1 = j], \tag{5.168}$$

respectively. Thus,

$$\sum_{j\in S}\alpha(j)v_{\lambda}^{d_{\mathbf{x}}^{\infty}}(j) = \sum_{s\in S}\sum_{a\in A_s} x_{d_{\mathbf{x}}}(s,a)r(s,a). \tag{5.169}$$

□

The objective functions for both the primal and dual LPs can be interpreted as the expected discounted reward of policy d^{∞}, in which the expectation is taken over the initial state distribution, in addition to over actions chosen by randomized decision rules and state transitions determined by transition probability function. Note the above result does not ensure that $\mathbf{v}_{\lambda}^{d_{\mathbf{x}}^{\infty}}$ is primal feasible. However, when $\mathbf{x}$ is optimal for the dual, then $\mathbf{v}_{\lambda}^{d_{\mathbf{x}}^{\infty}}$ will be optimal (and feasible, by definition) for the primal.

Optimal solutions and optimal policies

This section examines the relationship between optimal solutions to the dual linear program and it optimal policies.

Theorem 5.26. Let $\boldsymbol{\alpha} > \mathbf{0}$.

1. The dual linear program has an optimal solution.
2. Let $\mathbf{v}^*$ be an optimal solution to the primal linear program and $\mathbf{x}^*$ be an optimal solution to the dual linear program. Then

$$\sum_{s \in S} \alpha(s) v^*(s) = \sum_{s \in S} \sum_{a \in A_s} x^*(s,a) r(s,a). \tag{5.170}$$

Proof. The first result follows from Theorem 5.25 and Theorem C.1. The second result follows from Proposition 5.12. □

Theorem 5.26 is a restatement of the Theorem of Strong Duality of linear programming (Theorem C.6 in Appendix C) for discounted Markov decision processes. It is an immediate consequence of Theorem 5.22 that establishes that the primal LP has an optimal solution.

The next result formalizes the connection between optimal solutions to the dual LP and optimal policies of the MDP. This is the main result of this section.

Theorem 5.27.
1. Suppose $\mathbf{x}^*$ is an optimal solution to the dual LP. Then $d^\infty_{\mathbf{x}^*}$ is an optimal policy.
2. Suppose $(d^*)^\infty$ is an optimal policy. Then $\mathbf{x}_{d^*}$ is an optimal solution to the dual LP.

Proof. Let $\mathbf{v}^*$ and $\mathbf{x}^*$ be optimal solutions to the primal and dual LP, respectively. Let $d_{\mathbf{x}^*}$ be as defined in equation (5.153). Combining equation (5.169) and (5.170) gives

$$\sum_{s \in S} \alpha(s) v_\lambda^{d^\infty_{\mathbf{x}^*}}(s) = \sum_{s \in S} \sum_{a \in A_s} x^*(s,a) r(s,a) = \sum_{s \in S} \alpha(s) v^*(s).$$

For all s, since $\alpha(s) > 0$,

$$v_\lambda^{d^\infty_{\mathbf{x}^*}}(s) = v^*(s). \tag{5.171}$$

Since $\mathbf{v}^*$ is an optimal solution to the primal, it satisfies $\mathbf{v}^* \geq \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}^*$ for all $d \in D^{\mathrm{MD}}$, or equivalently, $\mathbf{v}^* \geq L\mathbf{v}^*$. By Theorem 5.7, $\mathbf{v}^* \geq \mathbf{v}^*_\lambda$. Hence,

$$v_\lambda^{d^\infty_{\mathbf{x}^*}}(s) \geq v^*_\lambda(s) \tag{5.172}$$

for all s. By the definition of $v^*_\lambda(s)$, (5.172) holds with equality so that $d^\infty_{\mathbf{x}^*}$ is an optimal policy.

To prove the second result, from (5.169) it follows that

$$\sum_{s \in S} \sum_{a \in A_s} x_{d^*}(s,a) r(s,a) = \sum_{s \in S} \alpha(s) v_\lambda^{(d^*)^\infty}(s)$$

$$\begin{aligned}
&\geq \sum_{s\in S} \alpha(s) v_\lambda^{d^\infty}(s) \\
&= \sum_{s\in S} \sum_{a\in A_s} x_d(s,a) r(s,a) \\
&= \sum_{s\in S} \sum_{a\in A_s} x(s,a) r(s,a),
\end{aligned}$$

for all $d \in D^{\mathrm{MR}}$. Since any feasible solution $\mathbf{x}$ to the dual represents a decision rule $d \in D^{\mathrm{MR}}$ for which $\mathbf{x}_d = \mathbf{x}$, no feasible solution has objective function value strictly greater than $\mathbf{x}_{d^*}$. Hence, $\mathbf{x}_{d^*}$ is an optimal solution to the dual LP. □

The following theorem is an immediate consequence of Theorem 5.27 applied to basic feasible solutions. It is also a direct consequence of Theorem 5.24.

Theorem 5.28. 1. Suppose $\mathbf{x}^*$ is an optimal basic feasible solution to the dual LP. Then $d^\infty_{\mathbf{x}^*}$ is a deterministic optimal policy.

2. Suppose $(d^*)^\infty$ is a deterministic optimal policy. Then $\mathbf{x}_{d^*}$ is an optimal basic feasible solution to the dual LP.

This theorem is of special note because:

1. Any linear program that has an optimal solution and whose feasible region has at least one basic feasible solution has an optimal basic feasible solution[36].
2. Earlier results establish the existence of an optimal deterministic stationary policy using properties of the Bellman equation.
3. When one exists, LP solvers will return an optimal basic feasible solution.

Some further results*

The dual LP can be used to provide the following direct proof of the existence of a stationary deterministic optimal policy in a discounted Markov decision process.

Since the feasible region includes non-negativity constraints, which means it does not contain a line[37], it follows that it has at least one vertex (see Theorem C.4). A fundamental theorem in linear programming states that if an LP has an optimal solution and the feasible region contains at least one vertex, then at least one of the vertices will

[36] Note that linear programs can have optimal solutions but no optimal basic feasible solution. For example, every point on the line $(x_1, 0)$ in $\mathbb{R}^2$ is optimal if the objective is to minimize x_2 subject to $x_2 \geq 0$, but none are basic feasible solutions. In the case of a Markov decision process, the dual LP polyhedron will always have extreme points since it is bounded.

[37] A set X *contains a line* if there exists a vector $\mathbf{x} \in X$ and a nonzero vector $\mathbf{d}$ such that $\mathbf{x} + \theta\mathbf{d} \in X$ for all $\theta \in \mathbb{R}$.

be an optimal solution. Since each vertex corresponds to a stationary deterministic policy, this argument establishes that there will always exist a deterministic stationary optimal policy.

Finally, recall that it was previously established that an optimal policy is optimal for all $\boldsymbol{\alpha} > \mathbf{0}$, using the primal LP. A proof using the dual follows.

Theorem 5.29. Let $\mathbf{x}^*$ be an optimal dual solution. The corresponding policy $d^\infty_{\mathbf{x}^*}$ is optimal for all $\boldsymbol{\alpha} > \mathbf{0}$.

Proof. First, consider the case where $\mathbf{x}^*$ is an optimal basic feasible solution. Let $d_{\mathbf{x}^*}$ be the policy corresponding to $\mathbf{x}^*$. Then, since $x^*(s,a) = 0$ for all $a \neq d_{\mathbf{x}^*}(s)$, $\mathbf{x}^*$ satisfies (5.151b):

$$(\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{x}^*}})^\mathsf{T} \mathbf{x}^* = \boldsymbol{\alpha}. \tag{5.173}$$

Since $\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{x}^*}}$ is invertible and since the transpose of the inverse is the inverse of the transpose,

$$\mathbf{x}^* = (\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{x}^*}})^{-1} \boldsymbol{\alpha} \geq \mathbf{0}, \tag{5.174}$$

where the inequality follows from Lemma 5.3. This result shows that $\mathbf{x}^*$ is feasible for all $\boldsymbol{\alpha} > \mathbf{0}$. Since changing $\boldsymbol{\alpha}$ does not affect the objective function, $\mathbf{x}^*$ is optimal for the dual LP and $d^\infty_{\mathbf{x}^*}$ is an optimal policy for the MDP for all $\boldsymbol{\alpha} > \mathbf{0}$.

In the case where $\mathbf{x}^*$ is an optimal solution that is not basic, then $\mathbf{x}^*$ can be written as the convex combination of optimal basic feasible solutions. By the above argument, for a given basic feasible solution, the same action is chosen in a given state for all $\boldsymbol{\alpha} > \mathbf{0}$. So the randomized policy corresponding to $\mathbf{x}^*$ will also be optimal for all $\boldsymbol{\alpha} > \mathbf{0}$. □

5.9.4 Solving linear programs

This section describes algorithms and solvers that can be used to find solutions to a linear programs.

Simplex algorithm

The simplex algorithm consists of two phases. Phase one of the simplex algorithm obtains a basic feasible solution to an LP. Phase two iteratively evaluates adjacent basic feasible solutions with improved objective function values until no improvement is possible. Since there are only finitely many basic feasible solutions, it must stop in a finite number of iterations with a local optimum. Since linear programs are convex optimization problems, finding a local optimum is equivalent to finding a global optimum.

Because basic feasible solutions of the dual LP correspond to stationary deterministic policies of the MDP, moving between adjacent basic feasible solutions means that improved actions are identified one state at a time. In other words:

> The simplex algorithm is equivalent to policy iteration in which the value function is updated after an improvement has been identified in a **single** state.

Conversely,

> Policy iteration in which the value function is updated after the improvement step cycles through all states is equivalent to a simplex algorithm with "block pivoting".

Block pivoting in simplex algorithms means that instead of obtaining the value of an improved *adjacent* vertex, the algorithm may jump to and evaluate a further away basic feasible solution, which represents a new deterministic stationary policy with improved actions in multiple states.

Figure 5.14 provides insight into this distinction. Suppose that the simplex algorithm starts at $\mathbf{x}^1$ and the optimal basic feasible solution is at $\mathbf{x}^2$. Then successive iterates would move through adjacent vertices and corresponding stationary policies to reach $\mathbf{x}^2$, while on the other hand, policy iteration might jump immediately to $\mathbf{x}^2$.

Interior point algorithms

Because the simplex algorithm repeatedly seeks and evaluates vertices, it is referred to as an exterior point algorithm. That is, it moves along the outside of the feasible region.

In contrast, interior point algorithms move towards an optimal basic feasible solution through the interior of the feasible region. That is, they approach an optimal deterministic policy through a sequence of randomized policies[38]. No Markov decision process algorithm is a direct analogue. Interior point methods have proven to be quite effective at solving many types of large linear programming problems, and so they may also be effective at solving Markov decision processes.

To visualize this distinction, consider Figure 5.14. Suppose the optimal basic feasible solution is at $\mathbf{x}^2$ and either algorithm starts iteration at $\mathbf{x}^1$. The simplex algorithm would iterate around the outside of the polyhedron while interior point methods would cut across the middle through points of the form $\bar{\mathbf{x}}$.

[38] There may be a relationship between interior point methods and policy gradient algorithms (Chapter 11), which improve over the class of stationary randomized policies.

Solvers

At the time of writing this book, the two most widely used commercial LP solvers are CPLEX (IBM [2024]) and Gurobi (Gurobi Optimization [2024]). They reliably solve large, industrial scale linear programs by exploiting modern computing architectures and encoding variants of the simplex and interior point algorithms.

Several open-source solvers are also available including GLPK (Makhorin [2024]) and COIN-OR CLP (Forrest and the COIN-OR contributors [2024]). These and other open-source solvers interface with all popular languages and applications.

Note that software that solves a primal (dual) linear program will also return the optimal values of its dual (primal) variables. This feature will be extremely useful when using the dual linear program to solve approximate dynamic programs.

5.10 Optimality of structured policies

This section extends results in Section 4.4 to infinite horizon models. The main result is stated in considerable generality.

5.10.1 The fundamental result

Define V^F to be a subset of V that contains elements of a specific form (such as convex) and define D^F to be a subset D^{MD} in which all decision rules are structured. They are *compatible* if $\mathbf{v} \in V^F$ implies that there exists a $d' \in \arg\text{c-max}_{d \in D^{\text{MD}}} L\mathbf{v}$ that is in D^F.

For example, in the queuing service rate control model (Section 3.4.1), if V^F denotes the set of convex non-decreasing functions then D^F is the set of monotone non-decreasing decision rules in D^{MD}.

The following theorem uses vector and operator notation. It generalizes Theorem 4.5 for finite horizon models to infinite horizon discounted models.

Theorem 5.30. Suppose that

1. V^F is non-empty,
2. $\mathbf{v} \in V^F$ implies $L\mathbf{v} \in V^F$, and
3. if $\lim_{n\to\infty} \|\mathbf{v}^n - \mathbf{v}^*\| = 0$ and $\mathbf{v}^n \in V^F$ for all $n = 1, 2, \ldots$ implies that $\mathbf{v}^* \in V^F$.

Then if D^F is compatible with V^F, there exists an optimal stationary policy $(d^*)^\infty$ with $d^* \in D^F$.

Proof. The first part of the proof is by induction. Since V^F is non-empty, let $\mathbf{v}^1 \in V^F$ be arbitrary and suppose that $\mathbf{v}^n \in V^F$ for $n = 1, \ldots, t$. Hypothesis 2 implies that

$\mathbf{v}^{t+1} = L\mathbf{v}^t \in V^F$. Hence $\mathbf{v} \in V^F$ for all n.

As a consequence of Theorem 5.16, $\mathbf{v}^n$ converges to $\mathbf{v}^*_\lambda$, the unique solution of the Bellman equation $L\mathbf{v} = \mathbf{v}$. Hence by Hypothesis 3, $\mathbf{v}^*_\lambda \in V^F$. Consequently there exists

$$d^* \in \arg\operatorname*{c\text{-}max}_{d \in D^{\mathrm{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}^*_\lambda\}$$

that is also in D^F. □

Some comments about this theorem follow:

1. The proof relies on the already demonstrated convergence of value iteration to the solution of the Bellman equation.

2. This result differs from that in the finite horizon case by requiring that the $\lim_{n\to\infty} \mathbf{v}^n$ retains the structure of all previous iterates. To establish this requires results from analysis on the limiting behavior of functions[39].

3. Note that the result does *not* say that *every* $d \in \arg\operatorname{c\text{-}max}_{d\in D^{\mathrm{MD}}}\{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}^*_\lambda\}$ is in D^F, only that there exists at least one that is in D^F.

5.10.2 A preventive maintenance example

This section illustrates Theorem 5.30 by applying it to determine the structure of an optimal policy in a stationary infinite horizon version of the preventive maintenance model of Section 4.4.2. The Markov decision process formulation is repeated to make this section self-contained.

As before, $S = \{0, 1, \ldots, M\}$ represents the state of the machine with higher state values representing greater deterioration and $A_s = \{0, 1\}$ denotes the set of available actions in each state. Assuming that the state of the machine is known at each decision epoch, the decision maker may either restore the machine (action 1) to state 0 at cost K or operate the machine as is (action 0) incurring a per-period operating cost of $f(s)$ where $f(s)$ is a non-decreasing function of s. Consequently operating the machine in a more deteriorated state costs more than operating it in a less deteriorated state. Assume further that restoring the machine takes one period so that a restored machine in state 0 is available at the start of the next period.

Define

$$p(j|s,a) = \begin{cases} p & j = s+1, a = 0, s \le M-1 \\ 1-p & j = s, a = 0, s \le M-1 \\ 1 & j = 0, a = 1, s \le M \\ 0 & \text{otherwise.} \end{cases}$$

[39] For example, if $S = [0, 1]$ and V^F denotes the set of continuous functions on S, then to establish the third result requires the result that the *uniform* limit of continuous functions is continuous.

Thus in each period the machine deteriorates one state with probability p and remains in the same state with probability $1-p$ except in state M. Since the decision maker seeks to minimize the expected discounted cost of operating the machine over the infinite horizon the problem is best expressed as a minimization problem.

The Bellman equation becomes

$$v(s) = \min\left\{K + \lambda v(0), f(s) + \lambda\big(pv(s+1) + (1-p)v(s)\big)\right\} \tag{5.175}$$

for $s \in \{0, \ldots, M\}$.

Existence of an optimal control limit policy

To apply Theorem 5.30, let V^F represent the set of non-decreasing real-valued functions on S. It was shown in Section 4.4.2 to be compatible with D^F, the set of control limit decision rules of the form:

$$d(s) = \begin{cases} 0 & \text{if } s \leq s^* \\ 1 & \text{if } s > s^*. \end{cases} \tag{5.176}$$

Clearly V^F is non-empty. Section 4.4.2 establishes Hypothesis 2 of the Theorem 5.30. Since the (uniform) limit of non-decreasing functions is non-decreasing, Hypothesis 3 holds. Consequently, there exists an optimal non-decreasing (control limit) policy. Moreover this result is valid if s is replaced with any non-decreasing function.

It is left as an exercise to verify this conclusion numerically, to develop an algorithm for finding an optimal control limit policy by searching within the class of control limit policies, and to provide conditions under which $s^* \in \{0, \ldots, M-1\}$.

5.11 Application: Queuing service rate control

This section compares numerical algorithms and investigates the structure of the optimal policy in the queuing service rate control model formulated in Section 3.4.1.

5.11.1 The model

Consider an infinite horizon discounted version of the discrete-time queuing service rate control model described in Section 3.4.1 and analyzed in the finite horizon case in Section 4.3.1. Assume three service probabilities $a_1 = 0.2$, $a_2 = 0.4$ and $a_3 = 0.6$ and an arrival probability of $b = 0.2$. Further assume that the delay cost is $f(s) = s^2$ and the cost per period of serving at rate a_k is $m(a_k) = 5k^3$.

To use a finite state algorithm requires truncation of the state space. Assume throughout this section that N denotes the truncation level. When the state space is truncated, the transition probabilities in state N for $k = 1, 2, 3$ are

$$p(j|N, a_k) = \begin{cases} a_k & j = N-1 \\ 1 - a_k & j = N. \end{cases}$$

This means that in state N, arrivals are blocked and the only possible event is a service completion.

Since the goal is to minimize costs, a formulation using "min" instead of "max" in the Bellman equation provides more transparent expressions. The Bellman equation for the truncated model follows:

$$v(s) = \min_{k=1,2,3} \begin{cases} m(a_k) + \lambda((1-b)v(0) + bv(1)) & s = 0 \\ f(s) + m(a_k) + \lambda(a_k v(s-1) + (1-b-a_k)v(s) + bv(s+1)) & 0 < s < N \\ f(N) + m(a_k) + \lambda(a_k v(N-1) + (1-a_k)v(N)) & s = N \end{cases}$$

When $m(a_k)$ is increasing in k, which is reasonable for this application, there is no need to minimize in state 0 since it will always be optimal to use the least expensive service rate (or turn off the server if that were possible). Hence the Bellman equation in state 0 simplifies to

$$v(0) = m(a_1) + \lambda((1-b)v(0) + bv(1)). \tag{5.177}$$

The beauty of this model is that as a result of the simple transition structure, there is no need to write out the entire transition matrix when implementing an iterative algorithm.

5.11.2 Value iteration

This section explores the use of value iteration to solve the queuing service rate control model. It terminates with an ϵ-optimal policy by using a span-based stopping criterion and estimates the value function using the lower bound extrapolation.

Iterates of value iteration (Algorithm 5.3) satisfy the following easy-to-code recursions:

$$\begin{aligned} v'(0) &= m(a_1) + \lambda((1-b)v(0) + bv(1)), \\ v'(s) &= \min_{k=1,2,3} \{f(s) + m(a_k) + \lambda(a_k v(s-1) + (1-b-a_k)v(s) + bv(s+1))\},\ 0 < s < N, \\ v'(N) &= \min_{k=1,2,3} \{f(N) + m(a_k) + \lambda(a_k v'(N-1) + (1-a_k)v(N))\}. \end{aligned}$$

The problem is solved for $N = 50$, 200 and 1,000, and $\lambda = 0.5$, 0.9 and 0.99. Each instance starts with $v'(s) = 0$ for all $s \in S$ as a starting value and $\epsilon = 0.0001$ for the stopping criterion. Value iteration terminates with an ϵ-optimal policy when

$$\text{sp}(\mathbf{v}' - \mathbf{v}) < \lambda(1-\lambda)^{-1}\epsilon.$$

At termination the value function is approximated by

$$\mathbf{v}_\lambda^\epsilon = \mathbf{v}' + \lambda(1-\lambda)^{-1}(\underline{\mathbf{v}' - \mathbf{v}})\mathbf{e},$$

which is within ϵ of the optimal value function. While value iteration is only guaranteed to produce an ϵ-optimal policy, analysis using policy iteration below shows that the policy obtained by value iteration is optimal.

Table 5.2 shows the number of iterations for value iteration to achieve the stopping criterion as a function of N and λ. Observe that the number of iterations is increasing in both N and λ. In all cases, the time to solve it on a MacBook Pro M3 was negligible.

N \ λ	0.5	0.9	0.99
50	27	157	384
200	31	202	754
1,000	36	240	1,983

Table 5.2: Iterations of value iteration required to achieve span-based stopping criterion for the discounted queuing service rate control as a function of N and λ.

Figure 5.15 shows the ϵ-optimal stationary policy as a function of λ for two choices of N. Observe that in both instances the ϵ-optimal policy is **monotone** in the service rate. That is, when the queue length is longer, the server should work faster in spite of the higher cost. Moreover, the states at which ϵ-optimal actions change for each value of λ remain the same for all choices of N.

In addition observe that in each state, the service probability increases with respect to λ. For example when $s = 15$, the left figure[40] shows that the ϵ-optimal action is $a_1 = 0.2$ for $\lambda = 0.5$, $a_2 = 0.4$ for $\lambda = 0.9$ and $a_3 = 0.6$ when $\lambda = 0.99$. The reason for this is that when the future becomes less important to the decision maker (that is when λ is smaller), it is not worth the extra cost of using a faster service rate to reduce the queue length. In the extreme, when $\lambda = 0.01$, the ϵ-optimal action in all states (verified up to $N = 5{,}000$) is a_1.

Figure 5.16 shows that the value function is non-decreasing in the queue length. Providing a good parametric approximation to this function will provide the basis for calculations in Chapter 9 using approximate dynamic programming.

5.11.3 Gauss-Seidel value iteration

This section finds ϵ-optimal policies using Gauss-Seidel value iteration (Algorithm 5.4). Its implementation updates states in sequential order as follows:

$$v(0) = m(a_1) + \lambda\big((1-b)v(0) + bv(1)\big),$$

$$v'(s) = \min_{k=1,2,3}\{f(s) + m(a_k) + \lambda\big(a_k v'(s-1) + (1-b-a_k)v(s) + bv(s+1)\big)\},\ 0 < s < N$$

[40] This observation is based on explicitly comparing the ϵ-optimal policies.

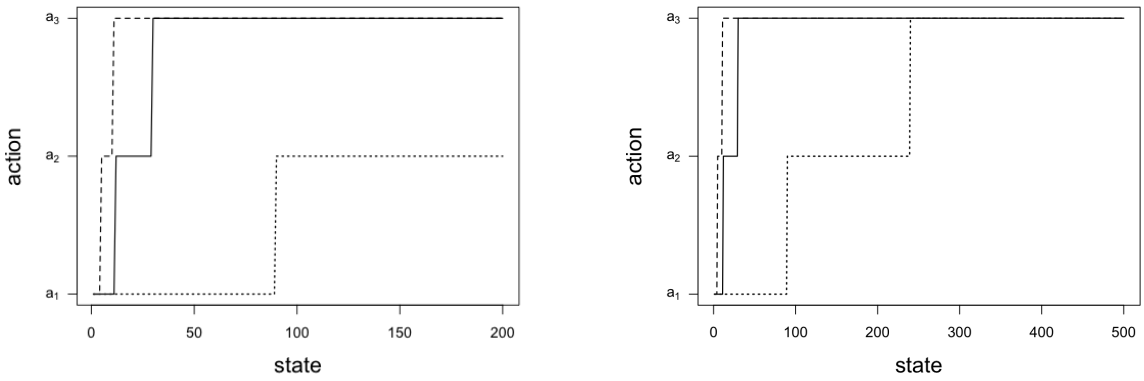


Figure 5.15: The ϵ-optimal stationary policy for the discounted queuing service rate control model as a function of λ for $\lambda = 0.5$ (dotted), $\lambda = 0.9$ (solid) and $\lambda = 0.99$ (dashed) truncated at $N = 200$ (left) and $N = 500$ (right).

$$v'(N) = \min_{k=1,2,3} \{f(N) + m(a_k) + \lambda\big(a_k v'(N-1) + (1-a_k)v(N)\big)\}.$$

Observe that these updates differ from value iteration in that they use $v'(s-1)$ instead of $v(s-1)$ for $s > 1$. As noted earlier, the $s = 1$ case for Gauss-Seidel and value iteration are identical.

Section 5.6.2 indicated that using a span-based stopping criterion for Gauss-Seidel iteration lacks theoretical justification[41]. Instead the Gauss-Seidel implementation uses the norm-based stopping criterion

$$\|\mathbf{v}' - \mathbf{v}\| < (1-\lambda)^{-1}\epsilon,$$

which by Theorem 5.18 guarantees that $\mathbf{v}'$ is within ϵ of the optimal value function and that a stationary policy derived using (5.114) or alternatively from a $\mathbf{v}$-greedy decision rule is an ϵ-optimal policy.

Table 5.3 shows that with the exception of the 50-state model with $\lambda = 0.99$, Gauss-Seidel requires fewer iterations than value iteration to identify an ϵ-optimal policy in spite of using the norm-based stopping criterion. At termination $\|\mathbf{v}'' - \mathbf{v}'\|/\|\mathbf{v}' - \mathbf{v}\|$, where $\mathbf{v}''$, $\mathbf{v}'$ and $\mathbf{v}$ denote three successive iterates, was slightly less than λ (0.987 vs. 0.99, 0.887 vs. 0.9 and 0.443 vs. 0.5), consistent with faster convergence.

5.11.4 Policy iteration

This section describes the use of policy iteration (Algorithm 5.6). The algorithm terminates when the $\mathbf{v}$-greedy decision rule found using (5.123) repeats or equivalently Γ

[41] An alternative is to intermittently perform a value iteration update, followed by evaluating the span of the change in values.

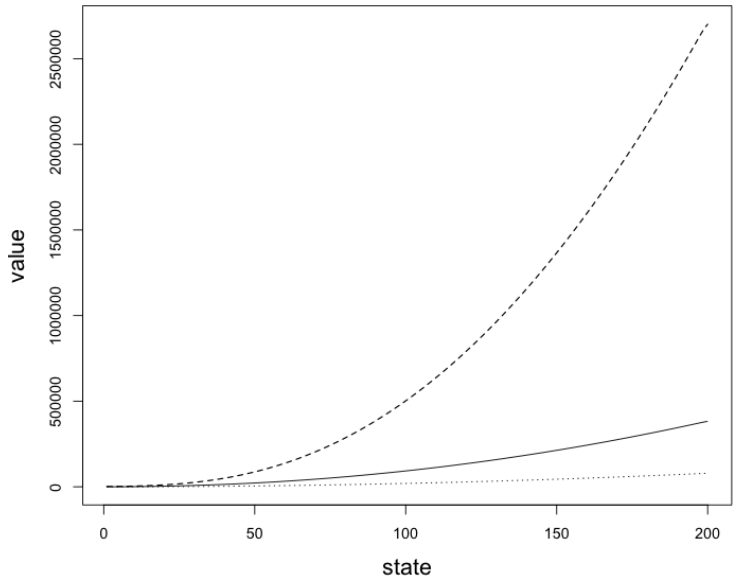


Figure 5.16: Lower bound extrapolation approximations to optimal value function for the discounted queuing service rate control model as a function of s for $\lambda = 0.5$ (dotted), $\lambda = 0.9$ (solid) and $\lambda = 0.99$ (dashed) truncated at $N = 200$.

N \ λ	0.5	0.9	0.99
50	23	90	634
200	24	90	634
1,000	24	102	727

Table 5.3: Iterations of Gauss-Seidel iteration required to achieve norm-based stopping criterion for the discounted queuing service rate control model for different values of N and λ.

is empty. In this implementation, policies are evaluated using the linear system solver in R Core Team [2024]. To do so requires generating the transition probability matrix corresponding to a decision rule, which in this model is tri-diagonal[42].

To explore the robustness of policy iteration, the algorithm is initialized with the sub-optimal decision rule $d^0 = \{a_1, a_2, a_3, a_1, a_2, a_3, \ldots\}$. The number of iterations required and the optimal policy are reported in Table 5.4.

Table 5.4 shows that in all instances except the first, three iterations (policy evaluations and minimizations) were required to find an optimal policy. Since the optimal

[42] A matrix is *tri-diagonal* if all non-zero entries lie on the sub-diagonal, diagonal and super-diagonal. The $row(\cdot)$ and $column(\cdot)$ functions in R Core Team [2024] facilitate generating this matrix.

N	λ	Iterations	First a_2	First a_3
50	0.50	2	n/a	n/a
50	0.90	3	11	29
50	0.99	3	4	10
200	0.50	3	89	n/a
200	0.90	3	11	29
200	0.99	3	4	10
1,000	0.50	3	89	239
1,000	0.90	3	11	29
1,000	0.99	3	3*	10

Table 5.4: Number of iterations and optimal policy obtained when solving the discounted queuing control model with policy iteration. Since the optimal policy is monotone in service rate, the column "First a_2" indicates the lowest state (shortest queue length) at which action a_2 (service rate 0.4) is optimal. Interpret the column "First a_3" analogously. The starred cell is discussed in the text and "n/a" indicates that this action was not used in the optimal policy.

policy for this model was monotone increasing in the service rate as a function of the queue length, the optimal policies can be described by the change points between actions as noted in the table caption. In all cases except one (denoted by *), the optimal policy for the smaller problem instance agreed with that of the larger problem instance.

The instance denoted by * is an anomaly arising from the dual effects of truncation and discreteness of the action space. When solving the problem with $N = 999$ and $N = 1001$, the optimal policy used a_1 for $s \leq 3$ and a_2 for $s \geq 4$ in agreement with the 50 and 200 state versions with $\lambda = 0.99$. When $N = 1{,}000$, it was observed that in the improvement step selecting the "argmax" using

$$a'_s \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_\lambda^{d^\infty}(j) \right\}$$

resulted in the quantities in brackets in state 4 being 2080.04 when $a = a_1$ and 2080.87 when $a = a_2$. Hence a_1 was selected when $N = 1{,}000$.

To further explore the performance of policy iteration and shed more light on the service rate control model, a larger problem instance was solved. In it, $A_s = \{a_1, \ldots, a_6\}$, corresponding to service rates $\{0.2, 0.3, \ldots, 0.7\}$. The model parameters were $m(a_k) = 2k^3$, $f(s) = s^2$, $b = 0.2$ and $N = 5{,}000$. The discount rate was set at $\lambda = 0.4$[43].

Policy iteration again found an optimal policy in 3 iterations, starting from the decision rule $d^0 = (a_1, a_6, a_1, a_6, \ldots)$. Solving the $5{,}000 \times 5{,}000$ linear system of equations

[43] A small discount rate was chosen in order to obtain a policy in which the change in optimal actions was not restricted to low states. With $\lambda = 0.9$, it was optimal to use action a_6 when $s \geq 106$.

required about 11 seconds on a MacBook Pro M3 computer using the solver in R Core Team [2024]. Figure 5.17 shows that the optimal policy is monotone and the value function is non-decreasing in the queue length.

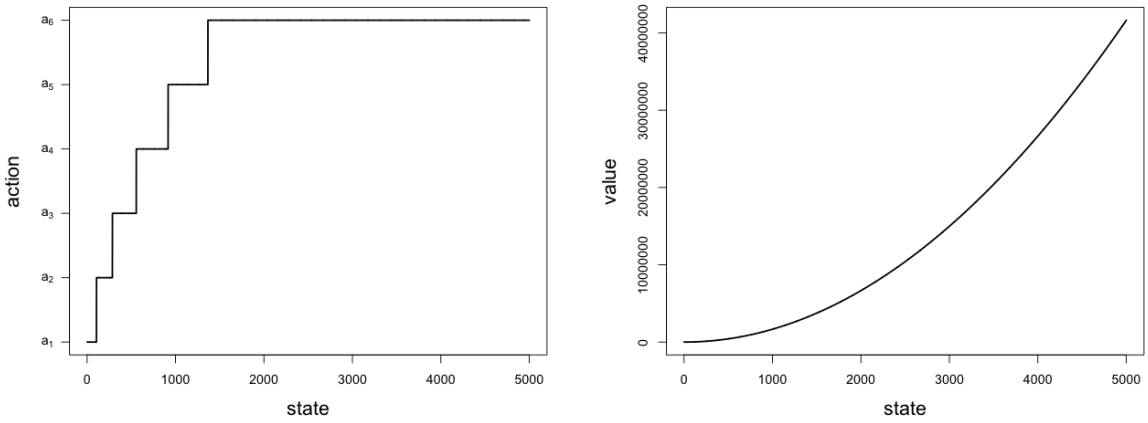


Figure 5.17: Optimal policy (left) and value function (right) obtained when using policy iteration to solve the larger instance of the service rate control problem.

5.11.5 Modified policy iteration

This section solves several versions this model using modified policy iteration with the main objectives being:

1. comparing the computational effort of MPI to that of value iteration,
2. comparing computation time of MPI to policy iteration in large problems, and
3. exploring the effect of the truncation sequence m_n on computational effort.

Theorem 5.13 is used to obtain an ϵ-optimal policy at termination.

Implementation follows Algorithm 5.8. It starts with a decision rule d' and value $\mathbf{v} = \mathbf{0}$ and applies the evaluation recursion (5.138) represented by

$$\mathbf{v} \leftarrow \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} \tag{5.178}$$

m_n times. The improvement step, (5.139), implemented component-wise, generates both an updated decision rule d and $L_d\mathbf{v}$.

To compare different truncation sequences, note that one improvement step requires the same number of calculations as

$$\frac{\sum_{s \in S} |A_s|}{|S|}$$

evaluation steps (5.138). More formally, define the quantity *Effort* to represent the number of calculations required to perform one evaluation step. MPI trades off the number of "cheap" evaluation steps with "more expensive" improvement steps. In this example, $|A_s| = 3$ for all $s \in S$. Hence an improvement step requires three times more calculations than an evaluation step. This means that if MPI terminated after M iterations, then the total *Effort* is $\sum_{n=1}^{M} m_n + 3M$.

Table 5.5 summarizes *Effort* as a function of the truncation sequence for the instance when the state space is truncated at $N = 200$, $\lambda = 0.9$ and MPI is initialized with the sub-optimal decision rule $d' = (a_1, a_2, a_3, a_1, a_2, a_3, \ldots)$. It shows that all implementations of modified policy iteration required fewer equivalent calculations than value iteration ($m_n = 0$). Note than the number of iterates for value iteration differed from that in Table 5.2 because this instance used a smaller value of ϵ. Observe also that the best choice of a fixed truncation value was between 10 and 20. Moreover the increasing sequences n and $\text{int}(n^{0.5})$[44] required more computational effort than the decreasing truncation sequence $\max(30 - n, 0)$, which required the least.

Truncation sequence m_n	Number of iterations, M	*Effort* $= \sum_{n=1}^{M} m_n + 3M$
0 (value iteration)	221	663
1	111	444
5	37	296
10	21	273
15	14	252
20	11	253
n	21	273
$\text{int}(n^{0.5})$	43	311
$\max(30 - n, 0)$	8	234

Table 5.5: Comparison of *Effort* of modified policy iteration in the queuing service rate control model using various truncation sequences for the instance with $N = 200$ and $\lambda = 0.9$. Entries above the line used fixed m_n and those below the line used m_n that varied in n.

Guided by the above results, a 1,000 state version with $\lambda = 0.9$ was solved. The sequence $m_n = \max(30 - n, 0)$ required the lowest *Effort*, 255, and $M = 10$ iterations. In contrast, *Effort* for $m_n = 20$ was 299 and that of value iteration was 783.

Solution methods were compared for variants of the six-action model (solved in the policy iteration section above) with 5,000, 10,000 and 15,000 states. Algorithms were initialized with the decision rule $d' = (a_1, a_2, a_3, a_1, a_2, a_3, \ldots)$ and $\epsilon = 0.00001$. Value iteration required 293 iterations corresponding to an *Effort* of 1758. Moreover, it required 21.69 minutes, to obtain an ϵ-optimal solution. In contrast, modified policy

[44] $\text{int}(x)$ denotes the integer part of x.

iteration with $m_n = \max(30-n, 0)$ required 12 iterations or, equivalently, *Effort* of 366 to find an ϵ-optimal solution. In contrast to value iteration, execution time for modified policy iteration was 1.38 minutes and that for policy iteration was 2.40 minutes. Thus, both are preferred to value iteration in this example.

Table 5.6 provides execution time for policy iteration and modified policy iteration in larger problems. Observe that modified policy iteration required less time than policy iteration in all cases with the greatest reduction when $N = 15{,}000$. In all cases policy iteration required three iterations to find an optimal policy. The majority of the execution time was spent in the evaluation step solving

$$(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{r}_d.$$

Note that constructing the matrix $\mathbf{P}_d$ required considerable computational time. Because of the sparsity of the $\mathbf{P}_d$ matrix, computational time for modified policy iteration could be reduced if (5.178) was implemented component-wise to take into account the matrix structure.

	Policy iteration		Modified policy iteration
N	Total time	Evaluation time	Total time
5,000	2.40	2.00	1.38
10,000	18.92	17.34	16.17
15,000	71.76	67.03	28.23

Table 5.6: Comparison of computation times (in minutes) on a MacBook M3 Pro for policy iteration and modified policy iteration with truncation series varying with respect to problem size. Total time solving systems of linear equations in the evaluation step of policy iteration appears in the column "Evaluation time".

5.11.6 Linear Programming

This section provides the primal and dual formulation of the queuing service rate control model and solves a problem instance using the dual.

Let $\alpha(s) > 0, s \in \{0, 1, \ldots, N\}$ satisfy $\sum_{s=0}^{N} \alpha(s) = 1$ and set $c(s, a_k) = f(s) + m(a_k) = s^2 + 5k^3$.

Primal linear program

Since this is a cost minimization problem, it is natural to formulate the primal linear program as a maximization problem[45] as follows:

[45]The primal LP formulation corresponding to a cost minimization Markov decision process is

$$\text{maximize} \quad \sum_{s \in S} \alpha(s) v(s)$$

$$
\begin{aligned}
\text{maximize} \quad & \alpha(0)v(0) + \cdots + \alpha(N)v(N) \\
\text{subject to} \quad & v(0) - \lambda((1-b)v(0) + bv(1)) \le c(0,a_k),\ k = 1,2,3 \\
& v(s) - \lambda(a_k v(s-1) + (1-a_k-b)v(s) + bv(s+1)) \le c(s,a_k),\ s = 1,\ldots,N-1, k = 1,2,3 \\
& v(N) - \lambda(a_k v(N-1) + (1-a_k)v(N)) \le c(N,a_k),\ k = 1,2,3.
\end{aligned}
$$

Observe that there are $3(N+1)$ inequalities, $N+1$ unknowns and no non-negativity constraints. Moreover, the left hand side of the first constraint does not depend on a_k, so it only needs to hold for the least costly action, a_1 (see equation (5.177)). Moreover with a little algebra, the constraints can be simplified.

Dual linear program

As noted earlier, it is preferable to solve the problem through its dual (5.151). The dual formulation is given by:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{s=0}^{N}\sum_{k=1}^{3} c(s,a_k)x(s,a_k) \\
\text{subject to} \quad & \sum_{k=1}^{3} x(0,a_k) - \lambda\sum_{k=1}^{3}\big((1-b)x(0,a_k) + a_k x(1,a_k)\big) = \alpha(0), \\
& \sum_{k=1}^{3} x(s,a_k) - \lambda\sum_{k=1}^{3}\big(bx(s-1,a_k) + (1-a_k-b)x(s,a_k) + a_k x(s+1,a_k)\big) = \alpha(s), \\
& \qquad\qquad \text{for } s = 1,\ldots,N-1, \\
& \sum_{k=1}^{3} x(N,a_k) - \lambda\sum_{k=1}^{3}\big(bx(N-1,a_k) + (1-a_k)x(N,a_k)\big) = \alpha(N), \\
& x(s,a_k) \ge 0, \quad k = 1,\ldots,3,\ s = 0,\ldots,N,
\end{aligned}
$$

Observe that the dual is a minimization problem and has $3(N+1)$ variables, $N+1$ equalities and $3(N+1)$ non-negativity constraints.

Solution

Recall that every feasible solution of the dual corresponds to a randomized policy through (5.153). Moreover from Theorem 5.24, it follows that decision rules corresponding to basic feasible solutions form deterministic policies in which $d^*(s) = a$

$$\text{subject to} \quad v(s) - \lambda\sum_{j\in S} p(j|s,a)v(j) \le r(s,a), \quad a \in A_s, s \in S.$$

Alternatively, the LP problem could be formulated as as a minimization problem, but this would require setting rewards equal to negative costs.

when $x(s,a) > 0$. Since there is an optimal basic feasible solution to the dual, the optimal policy can be easily obtained from it.

Consider a problem instance with $N = 50$, $\lambda = 0.9$ and $\alpha(s) = 1/(N+1)$. Since there are $N+1$ equality constraints, an optimal basic feasible solution to the dual has $N+1$ positive variables and $2(N+1)$ variables equal to zero. Solving the dual using either the function "simplex" in the package "boot" or the function "lpSolve" in the package "linprog" in R , yielded a dual solution of the form:

$$x(s,a_k) > 0 \quad \text{when} \quad \begin{cases} k = 1 \text{ and } s = 0,\ldots,10 \\ k = 2 \text{ and } s = 11,\ldots,28 \\ k = 3 \text{ and } s = 29,\ldots,50. \end{cases}$$

By the above argument, the policy $(d^*)^\infty$ which uses

$$d^*(s) = \begin{cases} a_1, \text{ for } s = 0,\ldots,10 \\ a_2, \text{ for } s = 11,\ldots,28 \\ a_3, \text{ for } s = 29,\ldots,50. \end{cases}$$

is optimal and agrees with that found using policy iteration.

Other observations include:

1. the primal solution, which yielded the optimal value function $\mathbf{v}^*_\lambda$, was identical to that found using the iterative methods above,
2. the optimal primal and dual objective function values were equal (Theorem 5.26),
3. the optimal policy and primal value functions did not vary with $\alpha(s)$ (Theorem 5.29), and
4. larger instances of the problem could be solved easily.

5.11.7 Concluding remarks on queuing control application

This section solved variants of the queuing service rate control model using all of the algorithms in this chapter. A summary of results follow:

1. In all cases the algorithms terminated with monotone non-decreasing policies and value functions. Although modified policy iteration and value iteration found ϵ-optimal policies, they were actually optimal.
2. Policy iteration (and linear programming) are the only algorithms that guarantee optimal (as opposed to ϵ-optimal) policies. However policy iteration requires constructing $\mathbf{P}_d$ and solving a linear system, which is $O(|S|^3)$, so that it becomes time consuming as the problem size increases.

3. Gauss-Seidel iteration required fewer iterations than value iteration but improvements were not remarkable.

4. Modified policy iteration with any reasonable truncation sequence significantly outperformed value iteration. It appeared that a decreasing truncation sequence worked best.

5. Modified policy iteration (with truncation sequence $m_n = \max(30 - n, 0)$) was the fastest method for finding optimal policies in large instances ($N \geq 5{,}000$).

6. Linear programming required some effort to formulate and specialized software to solve. The dual linear program identifies optimal policies directly and the primal yields optimal value functions. An advantage is that it allows constraints to be included.

For large problems, modified policy iteration with a decreasing truncation sequence appeared to be the best approach. Moreover it can be enhanced by using Gauss-Seidel updates when evaluating a fixed policy, coding the updates component-wise so as to exploit the sparsity of the transition matrices, and perhaps augmenting it with action elimination. Policy iteration can be used for small problems but in such settings, any solution method should suffice.

5.12 Application: Clinical decision making

This section analyzes a numerical instance of an infinite horizon discounted variant of the liver transplantation model that was formulated in Section 3.6. Refer to that section for model components and details.

The main purposes of this example are:

1. to analyze a model with a complex transition structure,

2. to illustrate an insightful graphical description of an optimal policy in a multi-dimensional state space,

3. to provide an optimal policy with a clinically meaningful structure, and

4. to delve deeper into an innovative application.

In this application, discounting may be justified in one of two ways: (1) from the perspective that one minus the discount rate represents the probability that a patient dies on any day from a cause other than organ failure or (2) from the perspective that a day alive today is worth more than a potential day alive in the future.

5.12.1 The Bellman equation

Given the model components defined previously, the Bellman equation can be written as

$$v(h,l) = \max\Big\{R(h,l), \sum_{\substack{h'\in S_H\\ l'\in S_L}}\Big(\big(1+\lambda v(h',l')\big)\gamma(h'|h)\beta(l'|h) + \big(1+\lambda v(h',\Phi)\big)\gamma(h'|h)\phi(h) + \lambda v(\Delta)\delta(h)\Big)\Big\} \tag{5.179}$$

for $h \in S_H, l \in S_L$,

$$v(h,\Phi) = \sum_{\substack{h'\in S_H\\ l'\in S_L}}\Big(\big(1+\lambda v(h',l')\big)\gamma(h'|h)\beta(l'|h)+\big(1+\lambda v(h',\Phi)\big)\gamma(h'|h)\phi(h)+\lambda v(\Delta)\delta(h)\Big) \tag{5.180}$$

and $v(\Delta) = 0$[46].

In the maximization, the first term corresponds to accepting the offered organ (a_0) while the second term corresponds to the decision to not accept an organ and wait one period (a_1). The second term is composed of the sum of three expressions: the first corresponds to a transition to state (h', l'), the second to no organ being available and the third corresponding to death due to liver failure. Note further that in each of the first two expressions, the quantity 1 represents an additional day of life.

Observe also that the second term in the maximization is identical to $v(h, \Phi)$ because rejecting an organ results in the same transition probabilities as when there is no organ to transplant[47].

5.12.2 A numerical example

Without access to primary data[48] the implementation here chose parameters that produce an optimal policy with a similar structure to that in the article.

This example specified $H = 20$ and $L = 10$. The daily discount rate $\lambda = 0.99978$ corresponded to an annual discount rate of 0.99. To be compatible with discounting, $R(h, l)$ represents the average *discounted* survival time post transplant for a patient in health state h and organ quality l. It was set to

$$R(h,l) = \alpha\big(365(14 - 0.25(h-1) - 0.6(l-1))\big).$$

[46] Because Δ is a zero-reward absorbing state, this application could be modeled without discounting as a transient model as described in Chapter 6.

[47] This observation simplifies coding the algorithm. Note a similar structure was observed in the online dating model of Section 4.3.3.

[48] Recall this example is based on Alagoz et al. [2007].

With $\alpha = 0.15$, mean survival varies between 7 and 25.6 months[49]

Transition probabilities without transplant were chosen to be

$$\delta(20) = 0.001 \text{ and } \gamma(20|20) = 0.999$$

and for $h = 1, \ldots, 19$

$$\delta(h) = 0.0003h \quad \text{and} \quad \gamma(h'|h) = \begin{cases} 0.997 - 0.0004h & h' = h \\ 0.003 + 0.0001h & h' = h + 1. \end{cases}$$

These probabilities assume that each day, a patient may die, remain in the same health state, or transition to the next higher (worse) health state[50]. Note that properties of transient Markov chains (see Appendix B) can be used to compute the expected number of days to death without transplant by solving $(\mathbf{I} - \mathbf{Q})\mathbf{u} = \mathbf{1}$ where $\mathbf{Q}$ denotes the matrix with components $q(h'|h)$. Using this approach showed that expected survival times without transplant are between 3.3 months in health state 20 and 35.2 months in health state 1[51].

The probability of not being offered an organ for transplant, $\phi(h)$, depends on h, the health state at the start a day. To model the policy that patients in poorer health states are most likely to be offered an organ, the probability of a patient in health state not being offered an organ was chosen to be

$$\phi(h) = 1 - 0.01h \quad \text{for} \quad h = 1, \ldots, 20$$

and

$$\beta(l|h) = 0.001h \quad \text{for} \quad h = 1, \ldots, 20,\ l = 1, \ldots, 10.$$

This probability distribution assumes that the probability of being offered an organ on any day, varies between 0.01 and 0.2 and that organ quality varies uniformly over the quality classes.

5.12.3 Numerical solution

Numerical solutions are obtained by solving the model using policy iteration with iterative policy evaluation[52] and begin the iteration with the decision rule d' that uses

[49] In contrast, the literature suggests survival of 14 years when a low-health state patient receives a high quality organ. The multiplier of 0.15 on R provides interesting policies. With $\alpha = 1$, it is always optimal to transplant.

[50] When the probability of transitioning to worse health states or death increases as the patient's health state degrades, a property known as *increasing failure rate*

[51] Literature suggests that 3-month survival rates vary between 27% and 98% depending on patient health state (MELD score).

[52] This is equivalent to modified policy iteration with a very high order (20,000). Iterative evaluation was used for coding since it is tedious to generate and manipulate matrices with vector-valued states in R.

action a_1 (reject an offered organ) in every state (h, l) for $h \in S_H$ and $L \in S_L$. Note that under this policy, $v(h, l)$ does not depend on l so it can be written as $u(h)$. To find $u(h)$ solve

$$(\mathbf{I} - \lambda \mathbf{Q})\mathbf{u} = \mathbf{1} \tag{5.181}$$

where $\mathbf{I}$ denotes an $|S_H| \times |S_H|$ identity matrix, $\mathbf{Q}$ denotes an $|S_H| \times |S_H|$ matrix with elements $\gamma(h'|h)$[53] and $\mathbf{u}$ denotes an $|S_H|$-component vector with entries $u(h)$.

In Figure 5.18, $u(h)$ is depicted with a solid line. Observe that it declines from over 1056 days for a patient in health state 1 (the best health state) to under 99 days for a patient in health state 20 (the worst health state). This figure also compares survival without transplant to survival post-transplant for the best ($l = 1$) and worst ($l = 10$) organs. It suggests that if the best organ is available, it should be accepted in health states 3-20 and if the worst organ is available it should be accepted in health states 8-20[54].

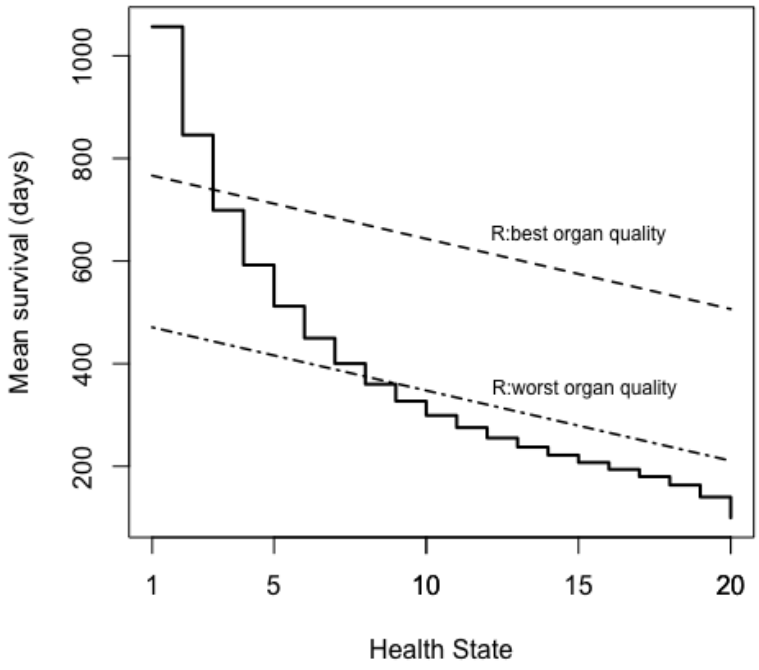


Figure 5.18: Comparison of mean (discounted) survival days $u(h)$ without transplant (solid line) and specified survival post transplant reward $R(h, l)$ for two organ qualities labelled "R:best organ quality" and "R:worst organ quality".

[53] This matrix corresponds to transient states of the health state change process.

[54] Note that choice of $R(h, l)$ was not based on data but specified to generate the pattern in Figure 5.18.

To implement the first improvement step of policy iteration set $v(h,l) = u(h)$ for $L = 1, \ldots, L$ and use the right hand side of (5.179) to identify an improvement. The algorithm converges after two additional iterations.

5.12.4 Interpretation of results

Figure 5.19a displays the optimal policy graphically. Observe that regardless of organ type, it is optimal to wait in health states 1 and 2 and to accept a transplant for any type of organ in health states 8-20. In health states 3-7, the decision to accept an organ depends on the organ quality. In health state 3, it is optimal to accept an organ of quality 1 and 2 while in health state 7 it is optimal to accept an organ of quality of 1 to 9. Observe also that the transplant region shrinks as health quality improves.

From the other perspective, it is optimal to accept the lowest quality organ in health states 8 to 20 and optimal to accept the best organ in health states 3 to 20. Observe also that the transplant region shrinks as the organ quality worsens.

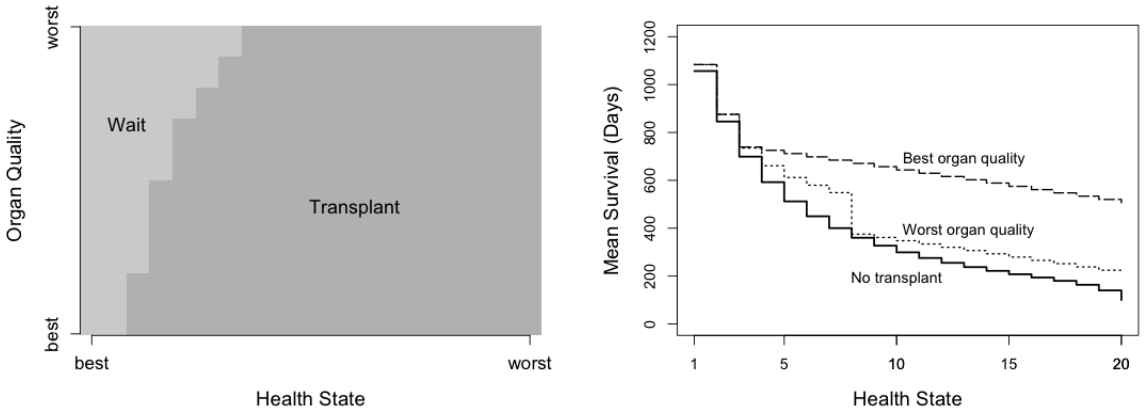
What this means clinically is that a patient in a poor health state should be willing to accept any available organ, while a patient in healthier state can be more selective. At the extremes, a patient in health states 1 or 2 should reject any organ and only consider transplantation when the disease worsens. In other words, a patient should become less selective as disease progresses. From a technical perspective, the optimal policy is a control limit policy in each dimension and jointly[55].

Figure 5.19b shows discounted survival times for the no transplantation option (as in Figure 5.18 and under the optimal policy when the best ($l = 1$) and worst ($l = 10$) organ types are available. Observe that survival times are greater than the always wait action in all health states under either transplant option. This may seem surprising in health state 1 since a patient in that state will not accept any organ. However after eventual transition to a poorer health state, the patient will follow the optimal policy and eventually accept a transplant. Thus improved survival follows because when the patient reaches a poorer health state, the improved survival under transplantation $R(h,l)$ will replace the survival under no transplantation $u(h)$.

5.12.5 Concluding remarks on clinical decision making application

This section has formulated and used Markov decision processes to analyze a stylized version of a realistic clinical decision problem. It provides a intuitive graphical representation of an optimal policy and interprets it from a clinical perspective. Moreover by comparing and interpreting survival (value functions) under several options, it shows the benefits of using a dynamic decision making as opposed to a myopic approach.

[55] Using methods discussed in Section 5.10, Alagoz et al. [2007] establish this result analytically under specific assumptions on the model components.



(a) Optimal policy as a function of health state and organ quality.

(b) Survival under the optimal policy as a function of health state for two organ qualities and without transplantation.

Figure 5.19: Graphical representation of the optimal policy and value function in the liver transplantation application.

5.13 Technical appendix

5.13.1 Proof of Theorem 5.1

This section proves Theorem 5.1 for the case where $v_\lambda^\pi(s)$ is written as:

$$v_\lambda^\pi(s) = \sum_{n=1}^{\infty} \sum_{j \in S} \sum_{a \in A_j} \lambda^{n-1} r(j,a,k) P^\pi[X_n = j, Y_n = a, X_{n+1} = k | X_1 = s]. \qquad (5.182)$$

Theorem 5.1 follows immediately from the following important yet subtle lemma[56]. Moreover this lemma will provide the basis for establishing an analogous result to Theorem 5.1 for models with the expected total reward and average reward criteria.

Given a history dependent randomized policy, the proof constructs for each starting state $s \in S$, a Markovian randomized policy with equivalent conditional action selection probabilities.

Lemma 5.9. Let $\pi = (d_1, d_2, \ldots) \in \Pi^{\text{HR}}$. For each $s \in S$ and $n = 1, 2, \ldots$, there exists a $\pi' = (d'_1, d'_2, \ldots) \in \Pi^{\text{MR}}$ for which

$$P^\pi[X_{n+1} = k, X_n = j, Y_n = a \,|X_1 = s] \qquad (5.183)$$
$$= P^{\pi'}[X_{n+1} = k, X_n = j, Y_n = a \,|X_1 = s]$$

[56] This result was proved by Strauch [1966] in greater generality.

for $k \in S$, $j \in S$, and $a \in A_j$.

Proof. Fix $s \in S$ and construct $\pi' = (d'_1, d'_2, \ldots) \in \Pi^{\text{MR}}$ by setting

$$w_{d'_n}(a|j,s) := P^\pi[Y_n = a \,|X_n = j, X_1 = s] = \frac{P^\pi[Y_n = a, X_n = j \,|X_1 = s]}{P^\pi[X_n = j \,|X_1 = s]} \tag{5.184}$$

for $j \in S$, $a \in A_j$ and $n = 1, 2, \ldots$[57].

Note the dependence on s in the definition of $w_{d'_n}(a|j,s)$. This is because the action selection probability distribution may differ across initial states. Note further the two quantities in the final term in (5.184) are marginal distributions over all intermediate states and actions[58].

The left hand side of (5.183) can be expressed as a product of conditional probabilities as follows:

$$\begin{aligned} &P^\pi[X_{n+1} = k, X_n = j, Y_n = a | X_1 = s] = \\ &\quad P^\pi[X_{n+1} = k | X_n = j, Y_n = a, X_1 = s]\, P^\pi[Y_n = a | X_n = j, X_1 = s]\, P^\pi[X_n = j | X_1 = s]. \end{aligned} \tag{5.185}$$

The following arguments establish that for $n = 1, 2, \ldots$, each expression in terms of $\pi \in \Pi^{\text{HR}}$ in (5.185) equals the equivalent expression with respect to $\pi' \in \Pi^{\text{MR}}$.

By the Markovian assumption on transition probabilities, X_{n+1} is independent of X_1, conditional on $X_n = j$ and $Y_n = a$. Therefore

$$P^\pi[X_{n+1} = k | X_n = j, Y_n = a, X_1 = s] = p(k|j,a) = P^{\pi'}[X_{n+1} = k | X_n = j, Y_n = a, X_1 = s].$$

By the construction of d'_n, it follows from (5.184) that

$$P^\pi[Y_n = a | X_n = j, X_1 = s] = w_{d'_n}(a|j,s) = P^{\pi'}[Y_n = a | X_n = j, X_1 = s] \tag{5.186}$$

for all n. Thus, what remains is to show is that

$$P^\pi[X_n = j | X_1 = s] = P^{\pi'}[X_n = j | X_1 = s]. \tag{5.187}$$

This follows by induction as follows. Clearly (5.187) holds for $n = 1$. Assume it is true for $t = 2, \ldots, n-1$. Then

$$\begin{aligned} P^\pi[X_n = j | X_1 = s] &= \sum_{l \in S} \sum_{a \in A_l} p(j|l,a) P^\pi[X_{n-1} = l, Y_{n-1} = a | X_1 = s] \\ &= \sum_{l \in S} \sum_{a \in A_l} p(j|l,a) P^\pi[Y_{n-1} = a | X_{n-1} = l, X_1 = s] P^\pi[X_{n-1} = l | X_1 = s] \\ &= \sum_{l \in S} \sum_{a \in A_l} p(j|l,a) P^{\pi'}[Y_{n-1} = a | X_{n-1} = l, X_1 = s] P^{\pi'}[X_{n-1} = l | X_1 = s] \end{aligned}$$

[57] The Markovian randomized decision rule d'_n induces a probability distribution over the actions that matches the probability distribution induced by the given history-dependent randomized decision rule d_n.

[58] See Example 5.19 below.

$$= \sum_{l \in S} \sum_{a \in A_l} p(j|l,a) P^{\pi'}[X_{n-1} = l, Y_{n-1} = a | X_1 = s]$$
$$= P^{\pi'}[X_n = j | X_1 = s],$$

where the third equality is a consequence of the induction hypothesis and (5.186). Hence (5.187) is valid for all n completing the proof. □

Substituting the result of the above lemma into (5.182) proves the following result that appears as Theorem 5.1 in the body of this chapter.

Theorem 5.31. For each $\pi \in \Pi^{\text{HR}}$, there exists a $\pi' \in \Pi^{\text{MR}}$ for which

$$v_\lambda^\pi(s) = v_\lambda^{\pi'}(s)$$

Comments and examples

Lemma 5.9 and its proof are quite subtle. Comments follow.

1. Given a history-dependent policy, this lemma guarantees that for each initial state s there exists a Markovian (randomized) policy $\pi'(s)$, that achieves the same state action probabilities. However, as Example 5.18 shows, it does not guarantee that same policy will achieve this result for all states.

2. The construction of d'_n in (5.184) requires some further explanation. The quantities $P^\pi[Y_n = a, X_n = j, |X_1 = s]$ and $P^\pi[X_n = j | X_1 = s]$ may be computed by *marginalizing* the distribution summing over all omitted terms, for example when deriving

$$P^\pi[X_2 = j \,|X_1 = s] = \sum_{a \in A_a} p(j|s,a) P^\pi[Y_1 = a \,|X_1 = s].$$

3. This result does not guarantee that distribution of sample paths of π and π' will be the same, only that the marginal conditional probabilities necessary to evaluate the expected reward in period n given in (5.183) agree.

4. The result generalizes to a model in which the initial state is chosen according to a probability distribution $\rho(s) := P[X_1 = s]$ in which the conditional action selection probabilities in (5.184) depend on $\rho(\cdot)$.

5. This result could also be applied to a finite horizon model to establish that for every history-dependent policy there exists a Markovian randomized policy with the same state and action probabilities.

Example 5.18. To show that the corresponding Markovian policy depends on the initial system state, consider a deterministic model with $S = \{u, w\}$ and $A_u = A_w = \{a, b\}$. Action a maintains the system in its current state and action b causes the system to move to the other state.

Define the history-dependent policy π that uses a twice and then b once and repeats this sequence indefinitely when the system starts in u and uses b once then uses a twice and repeats this sequence indefinitely when the system starts in w. Since the system is deterministic the action at decision epochs when a state is not visited can be arbitrary.

Hence if the system starts in state u, π generates the sequence of states and actions (history) $(u, a, u, a, u, b, w, a, w, a, w, b, u, \ldots)$ and if it starts in w, it generates the sequence of states and actions $(w, b, u, a, u, a, u, b, w, a, w, a, w, b, \ldots)$. Now construct the decision rules corresponding to a (non-stationary) Markovian policy. Starting in u, $d_1(u) = a$, $d_2(u) = a$ and $d_3(u) = b$. Starting in w, $d'_1(w) = b$, $d'_2(u) = a$ and $d'_3(u) = a$. Since $d_3(u) \neq d'_3(w)$, the Markovian decision rule needed to match the state-action distribution of π varies with the starting state.

The following numerical example illustrates the computation implicit in (5.184).

Example 5.19. Let $S = \{u, w\}$ and $A_u = A_w = \{a, b\}$, $p(u|u, b) = 0.1$ and $p(u|u, a) = 0.4$.

Suppose $\pi \in \Pi^{\mathrm{HR}}$ is such that

$$P^\pi[Y_1 = a|X_1 = u] = 0.7$$

and

$$P^\pi[Y_2 = a|X_2 = u, Y_1 = a, X_1 = u] = 0.2$$
$$P^\pi[Y_2 = a|X_2 = u, Y_1 = b, X_1 = u] = 0.4.$$

Note that at the second decision epoch, the probability distribution of action selection under π depends on the entire history.

Then for example

$$P^\pi[Y_2 = a, X_2 = u, Y_1 = a|X_1 = u] = 0.2 \times 0.4 \times 0.7 = 0.056$$
$$P^\pi[Y_2 = a, X_2 = u, Y_1 = b|X_1 = u] = 0.4 \times 0.1 \times 0.3 = 0.012$$

so that

$$P^\pi[Y_2 = a, X_2 = u|X_1 = u] = 0.056 + 0.012 = 0.068$$
$$P^\pi[X_2 = u|X_1 = u] = 0.4 \times 0.7 + 0.1 \times 0.3 = 0.31$$

Hence d_2' chooses actions according to

$$w_{d_2'}(a|u,u) = \frac{0.068}{0.31} = 0.219 \text{ and } w_{d_2'}(b|u,u) = \frac{0.224}{0.31} = 0.781$$

where $w_{d_2'}(\cdot|u,u)$ is defined for $n=2$ by (5.184). Thus at decision epoch 2, π' chooses actions according to d_2' and is in Π^{MR}.

5.13.2 Bounds for Gauss-Seidel iteration

This section generalizes arguments in Section 5.5.3 to derive bounds for Gauss-Seidel iteration. To do so requires developing some interesting matrix theory.

Choose $\mathbf{v} \in V$ and let $d' \in D^{\text{MD}}$ be such that for $k = 1, \ldots, M$

$$r(s, d'(s_k)) + \lambda \left(\sum_{j<k} p\big(s_j|s_k, d'(s_k)\big) G\mathbf{v}(s_j) + \sum_{j \geq k} p\big(s_j|s_k, d'(s_k)\big) v(s_j) \right)$$
$$= \max_{a \in A_s} \left\{ r(s,a) + \lambda \left(\sum_{j<k} p(s_j|s_k,a) G\mathbf{v}(s_j) + \sum_{j \geq k} p(s_j|s_k,a) v^n(s_j) \right) \right\} = G\mathbf{v}(s_k).$$

Decompose $\mathbf{P}_{d'}$ in to its lower and upper triangular parts, $\mathbf{P}_{d'} = \mathbf{P}_{d'}^L + \mathbf{P}_{d'}^U$, where

$$\mathbf{P}_{d'}^L = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ p_{21} & 0 & 0 & \ldots & 0 & 0 \\ p_{31} & p_{32} & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{M-1,1} & p_{M-1,2} & p_{M-1,3} & \ldots & \ddots & 0 \\ p_{M1} & p_{M2} & p_{M3} & \ldots & p_{M,M-1} & 0 \end{bmatrix}$$

and

$$\mathbf{P}_{d'}^U = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \ldots & p_{1,M-1} & p_{1M} \\ 0 & p_{22} & p_{23} & \ldots & p_{2,M-1} & p_{2M} \\ 0 & 0 & p_{33} & \ldots & p_{3,M-1} & p_{3M} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \ddots & p_{M-1,M} \\ 0 & 0 & 0 & \ldots & 0 & p_{MM} \end{bmatrix}.$$

Then in vector form

$$G\mathbf{v} = \underset{d \in D^{\text{MD}}}{\text{c-max}} \left\{ \mathbf{r}_d + \lambda \mathbf{P}_d^L G\mathbf{v} + \lambda \mathbf{P}_d^U \mathbf{v} \right\} = \mathbf{r}_{d'} + \lambda \mathbf{P}_{d'}^L G\mathbf{v} + \lambda \mathbf{P}_{d'}^U \mathbf{v}.$$

From the above equation, it is easy to see that if $\mathbf{P}_d$ is upper-triangular, that is $\mathbf{P}_d^L$ is a matrix of zeroes, Gauss-Seidel and value iteration are identical. Rearranging terms

in the above expression yields

$$(\mathbf{I} - \lambda \mathbf{P}_{d'}^L) G\mathbf{v} = \mathbf{r}_{d'} + \lambda \mathbf{P}_{d'}^U \mathbf{v}.$$

Since $(\mathbf{I} - \lambda \mathbf{P}_{d'}^L)^{-1}$ exists, it provides the following elegant representation for G

$$\begin{aligned} G\mathbf{v} &= (\mathbf{I} - \lambda \mathbf{P}_{d'}^L)^{-1} \mathbf{r}_{d'} + (\mathbf{I} - \lambda \mathbf{P}_{d'}^L)^{-1} \lambda \mathbf{P}_{d'}^U \mathbf{v} \\ &= \underset{d \in D^{\mathrm{MD}}}{\text{c-max}} \left\{ (\mathbf{I} - \lambda \mathbf{P}_d^L)^{-1} \mathbf{r}_d + (\mathbf{I} - \lambda \mathbf{P}_d^L)^{-1} \lambda \mathbf{P}_d^U \mathbf{v} \right\}. \end{aligned} \tag{5.188}$$

The following additional notation simplifies exposition and suggests some generalizations. For $d \in D^{\mathrm{MD}}$, define two matrices as follows:

$$\mathbf{Q}_d := \mathbf{I} - \lambda \mathbf{P}_d^L \quad \text{and} \quad \mathbf{R}_d := \lambda \mathbf{P}_d^U.$$

Consequently[59]

$$\mathbf{I} - \lambda \mathbf{P}_d = \mathbf{Q}_d - \mathbf{R}_d. \tag{5.189}$$

Hence G can be expressed in this notation as:

$$G\mathbf{v} = \underset{d \in D^{\mathrm{MD}}}{\text{c-max}} \{ \mathbf{Q}_d^{-1} \mathbf{r}_d + \mathbf{Q}_d^{-1} \mathbf{R}_d \mathbf{v} \}. \tag{5.190}$$

The following result generalizes Lemma 5.5 to the operator G. The proof is identical to that lemma in which $L\mathbf{v}$ is replaced by the representation (5.190) for $G\mathbf{v}$.

Lemma 5.10. Let $\mathbf{u} \in V$ and $\mathbf{u} \in V$. If $d_{\mathbf{v}}$ attains the maximum of $G\mathbf{v}$ in (5.190), then

$$G\mathbf{v} - \mathbf{v} \geq G\mathbf{u} - \mathbf{u} + (\mathbf{Q}_{d_{\mathbf{v}}}^{-1} \mathbf{R}_{d_{\mathbf{v}}} - \mathbf{I})(\mathbf{u} - \mathbf{v}). \tag{5.191}$$

An additional result follows.

Lemma 5.11. For any $d \in D^{\mathrm{MR}}$, $\|\mathbf{Q}_d^{-1} \mathbf{R}_d\| = \lambda$ and

$$(\mathbf{I} - \mathbf{Q}_d^{-1} \mathbf{R}_d)^{-1} = \sum_{n=0}^{\infty} (\mathbf{Q}_d^{-1} \mathbf{R}_d)^n. \tag{5.192}$$

Proof. From Lemma 5.2,

$$\mathbf{Q}_d^{-1} = (\mathbf{I} - \lambda \mathbf{P}_d^L)^{-1} = \sum_{n=0}^{\infty} (\lambda \mathbf{P}_d^L)^n \geq \mathbf{I}.$$

[59] This decomposition of $\mathbf{I} - \lambda \mathbf{P}_d$ is referred to as a *splitting*. Other iterative schemes for solving $(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{r}_d$ can be expressed in this way.

Since $(\mathbf{I} - \lambda\mathbf{P}_d)\mathbf{e} \geq \mathbf{0}$, the above inequality implies

$$\mathbf{Q}_d^{-1}(\mathbf{I} - \lambda\mathbf{P}_d)\mathbf{e} \geq (\mathbf{I} - \lambda\mathbf{P}_d)\mathbf{e},$$

and it follows from (5.189) that

$$(\mathbf{I} - \mathbf{Q}_d^{-1}\mathbf{R}_d)\mathbf{e} = \mathbf{Q}_d^{-1}(\mathbf{Q}_d - \mathbf{R}_d)\mathbf{e} \geq (\mathbf{I} - \lambda\mathbf{P}_d)\mathbf{e}.$$

Hence $\lambda\mathbf{P}_d\mathbf{e} \geq \mathbf{Q}_d^{-1}\mathbf{R}_d\mathbf{e} \geq \mathbf{R}_d\mathbf{e} \geq \mathbf{0}$, from which it follows that $\lambda = \|\lambda\mathbf{P}_d\| \geq \|\mathbf{Q}_d^{-1}\mathbf{R}_d\|$. But since the first row of Gauss-Seidel is equivalent to value iteration, its row sum, $\mathbf{Q}_d^{-1}\mathbf{R}_d\mathbf{e}(s_1) = \lambda$ from which the first result follows.

Equation (5.192) follows the first result and Lemma 5.2. □

This lemma is used in the proof of the following theorem but also guarantees the row sums of $\mathbf{Q}_d^{-1}\mathbf{R}_d$ are less than or equal to λ. If $\mathbf{P}_d$ is not upper triangular, some, but not necessarily all, of the row sums will be strictly less than λ.

For a matrix $\mathbf{A} \geq \mathbf{0}$, let $\mu(\mathbf{A})$ denote its minimum row sum. Of course $\|\mathbf{A}\|$ equals its maximum row sum. Note that the proof of the following theorem differs slightly from that of Theorem 5.12 because the row sums of $\mathbf{Q}_d^{-1}\mathbf{R}_d$ need not all equal λ.

Theorem 5.32. Suppose $0 \leq \lambda < 1$ and let $\alpha = \min_{d \in D^{\text{MD}}} \mu(\mathbf{Q}_d^{-1}\mathbf{R}_d)$. Then for any $\mathbf{v} \in V$,

$$\begin{aligned}\mathbf{v} + (1-\lambda)^{-1}(\overline{G\mathbf{v} - \mathbf{v}})\mathbf{e} \geq G\mathbf{v} + \lambda(1-\lambda)^{-1}(\overline{G\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v}_\lambda^* \geq \mathbf{v}_\lambda^{d_\mathbf{v}^\infty} \\ \geq G\mathbf{v} + \alpha(1-\alpha)^{-1}(\underline{G\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v} + (1-\alpha)^{-1}(\underline{G\mathbf{v} - \mathbf{v}})\mathbf{e}\end{aligned} \tag{5.193}$$

Proof. Following the argument used to obtain the upper bound in the proof of Theorem 5.12 and (5.192), it follows that

$$\mathbf{v} + \sum_{n=0}^{\infty}(\mathbf{Q}_{d_\mathbf{v}}^{-1}\mathbf{R}_{d_\mathbf{v}})^n(\overline{G\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v} + (\mathbf{I} - \mathbf{Q}_{d_\mathbf{v}}^{-1}\mathbf{R}_{d_\mathbf{v}})^{-1}(G\mathbf{v} - \mathbf{v}) \geq \mathbf{v}_\lambda^*.$$

From Lemma 5.11, $\|\mathbf{Q}_{d_\mathbf{v}}^{-1}\mathbf{R}_{d_\mathbf{v}}\| = \lambda$ so that sum above is bounded by $(1-\lambda)^{-1}$ giving the outer upper bound. The inner upper bound follows from the same argument used to prove Theorem 5.12.

To obtain the lower bound, apply Lemma 5.10 and (5.192) to obtain

$$\mathbf{v}_\lambda^* \geq \mathbf{v} + \sum_{n=0}^{\infty}(\mathbf{Q}_{d_\mathbf{v}}^{-1}\mathbf{R}_{d_\mathbf{v}})^n(\underline{G\mathbf{v} - \mathbf{v}})\mathbf{e}.$$

Follow the steps in the proof of Theorem 5.12 using the lower bounds on $\sum_{n=0}^{\infty}(\mathbf{Q}_{d_\mathbf{v}}^{-1}\mathbf{R}_{d_\mathbf{v}})^n\mathbf{e}$ based on α to complete the proof. □

Note that the upper bounds can be computed throughout the iterative process. However the lower bound cannot be evaluated without knowledge of α or a lower bound on it. When $\|G\mathbf{v} - \mathbf{v}\|$ is sufficiently small to terminate the iterations, the upper bound should provide a reasonable approximation to the optimal value function.

The following example illustrates these bounds in a case where α can be evaluated.

Example 5.20. Consider to the two-state example with $\lambda = 0.9$. Since there are only four Markovian decision rules, one can evaluate α. The following table summarizes calculations.

d	$\mathbf{Q}_d$	$\mathbf{R}_d$	$\mathbf{Q}_d^{-1}\mathbf{R}_d$	$\|\mathbf{Q}_d^{-1}\mathbf{R}_d\|$	$\mu(\mathbf{Q}_d^{-1}\mathbf{R}_d)$
$(a_{1,1}, a_{2,1})$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.72 & 0.8 \\ 0 & 0.9 \end{bmatrix}$	$\begin{bmatrix} 0.72 & 0.18 \\ 0 & 0.9 \end{bmatrix}$	0.9	0.9
$(a_{1,1}, a_{2,2})$	$\begin{bmatrix} 1 & 0 \\ -0.36 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.72 & .18 \\ 0 & 0.54 \end{bmatrix}$	$\begin{bmatrix} 0.72 & 0.8 \\ 0.26 & 0.60 \end{bmatrix}$	0.9	0.86
$(a_{1,2}, a_{2,1})$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.9 \\ 0 & 0.9 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.9 \\ 0 & 0.9 \end{bmatrix}$	0.9	0.9
$(a_{1,2}, a_{2,2})$	$\begin{bmatrix} 1 & 0 \\ -0.36 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.9 \\ 0 & 0.54 \end{bmatrix}$	$\begin{bmatrix} 0 & 0.9 \\ 0 & 0.54 \end{bmatrix}$	0.9	**0.54**

Observe that $\|\mathbf{Q}_d^{-1}\mathbf{R}_d\| = 0.9$ as guaranteed by Lemma 5.11. Also note that $\alpha = 0.54$. Since α corresponds to the optimal policy, one would expect the lower bound to apply.

The following table of iterates of Gauss-Seidel with $\mathbf{v}^0 = \mathbf{0}$ shows how these bounds behave empirically. Each entry of the table represents a bound on each component of the vector of iterates.

Iteration	Upper bound	Lower bound
10	$(32.958, 31.022)$	$(24.597, 22.661)$
20	$(30.799, 28.655)$	$(28.860, 26.717)$
30	$(30.299, 28.107)$	$(29.849, 27.657)$
40	$(30.182, 27.980)$	$(30.078, 27.875)$

Since $\mathbf{v}^*_\lambda = (30.147, 27.941)$, the bounds are quite adequate after 40 iterations with the upper bound being a closer approximation of $\mathbf{v}^*_\lambda$ than the lower bound. Note also that when α replaced λ in the upper bound and λ by α in the lower bound, neither were valid.

5.13.3 Convergence of policy iteration in non-finite models*

This section represents a departure from most of the book in that it applies to models with non-finite state and/or action sets. Figure 5.20 symbolically represents the logic underlying the Newton's method representation for policy iteration, suggests why it converges in a few iterations in a finite model and also provides the basis for proofs of convergence of modified policy iteration.

Iteratively applying Lemma 5.6 and using the established convergence of value iteration yields the following important convergence result for policy iteration. See the proof of Theorem 5.20 for a detailed proof using the same logic.

Theorem 5.33. Suppose $B\mathbf{v}^0 \geq \mathbf{0}$ and for all $\mathbf{v} \in V$ there exists a

$$d_{\mathbf{v}} \in \arg\max_{d \in D^{\mathrm{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}. \tag{5.194}$$

Then the sequence of iterates $\mathbf{v}^n, n = 0, 1, \ldots$ of policy iteration converges monotonically and in norm to $\mathbf{v}^*_\lambda$.

The condition $B\mathbf{v}^0 \geq \mathbf{0}$, or equivalently $L\mathbf{v}^0 \geq \mathbf{v}^0$, holds when policy iteration starts with a decision rule d^0 in step 2, which is evaluated in step 3 to obtain $\mathbf{v}^0$, because

$$L\mathbf{v}^0 \geq \mathbf{r}_{d^0} + \lambda \mathbf{P}_{d^0} \mathbf{v}^0 = B\mathbf{v}^0.$$

Alternatively policy iteration can start in step 3 with any $\mathbf{v}^0$ satisfying $B\mathbf{v}^0 \geq \mathbf{0}$. Recall that if $B\mathbf{v}^0 \geq \mathbf{0}$, $\mathbf{v}^0 \leq \mathbf{v}^*_\lambda$.

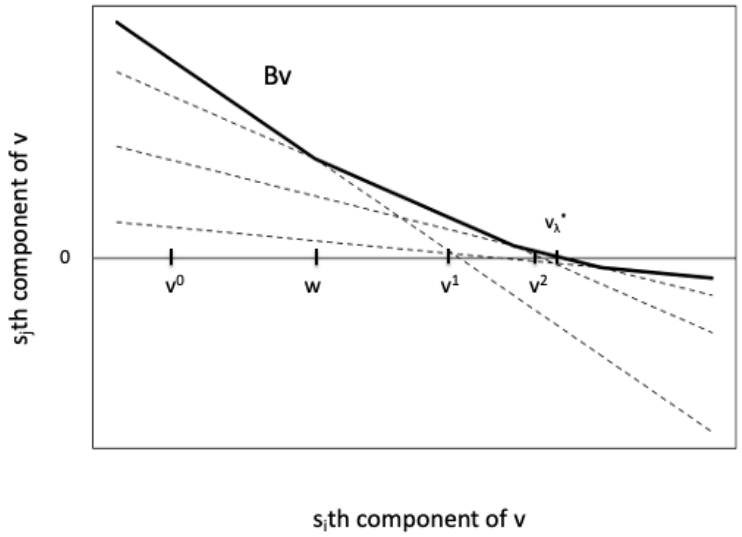


Figure 5.20: The bold line represents $B\mathbf{v}(s_j)$ as a function of $v(s_i)$ for two states s_i and s_j and the horizontal line represents $v(s_j) = 0$. The dashed lines represent $B_d\mathbf{v}(s_j)$ as a function of $v(s_i)$ for each of four decision rules. When policy iteration starts at $\mathbf{v}^0$, the improvement step identifies a $\mathbf{v}^0$-greedy decision rule d^0 with support $B_{d^0}\mathbf{v}$. In the evaluation step, it solves $B_{d^0}\mathbf{v} = \mathbf{0}$ to obtain the value $\mathbf{v}^1$. Repeating this process shows that policy iteration generates the iterates $\mathbf{v}^2$ and $\mathbf{v}^3 = \mathbf{v}^*_\lambda$. Observe that if one were able to identify $\mathbf{w}$ with less effort than solving $B_{d^0}\mathbf{v} = \mathbf{0}$, the $\mathbf{w}$-greedy decision rule would be identical to that at $\mathbf{v}^1$. Modified policy iteration seeks to do this.

The crucial hypothesis is that the maximum is attained in (5.194). Conditions under which this occurs include:

1. A_s finite for all $s \in S$, and

2. A_s compact for all $s \in S$, and $r(s,a)$ and $p(j|s,a)$ continuous in a for each $s \in S$.

Note that S no longer needs to be finite. It can also be compact or countable, and convergence will be guaranteed.

In non-finite circumstances, policy iteration may never satisfy the stopping criterion $d' = d$. In such cases, terminate the algorithm using a stopping criterion based on $\|\mathbf{v}^{n+1} - \mathbf{v}^n\|$ or $\text{sp}(\mathbf{v}^{n+1} - \mathbf{v}^n)$. The value of the decision rule at termination provides the tightest lower bound while the tightest upper bound from Theorem 5.12 applies.

Note that this approach can also be used when the problem is finite if one seeks only an ϵ-optimal policy. Note further that action elimination can also be incorporated in policy iteration.

Quadratic convergence of policy iteration

As observed in examples, policy iteration usually converges in a few iterations in finite state and action models. The following theorem provides conditions under which policy iteration converges quadratically in non-finite models.

Theorem 5.34. Suppose for all $\mathbf{u}$ and $\mathbf{v}$ in V, there exists a $K > 0$ for which

$$\|\mathbf{P}_{d_\mathbf{u}} - \mathbf{P}_{d_\mathbf{v}}\| \leq K\|\mathbf{u} - \mathbf{v}\| \tag{5.195}$$

where $d_\mathbf{v}$ and $d_\mathbf{u}$ are greedy with respect to $\mathbf{u}$ and $\mathbf{v}$. Then

$$\|\mathbf{v}^{n+1} - \mathbf{v}^*_\lambda\| \leq \frac{K\lambda}{1-\lambda}\|\mathbf{v}^n - \mathbf{v}^*_\lambda\|^2. \tag{5.196}$$

Refer to Section 6.4.4 in Puterman [1994] for a proof of this theorem, an example illustrating quadratic convergence and conditions on the model components that ensure (5.195) holds.

Note that condition (5.195) does not hold for all $\mathbf{u}$ and $\mathbf{v}$ in finite state and action models. To understand why consider the value $\mathbf{w}$ in Figure 5.20 at which there is a "kink" in $B\mathbf{v}$. This means that at $\mathbf{w}$ there are two $\mathbf{w}$-greedy decision rules, say d_1 and d_2, where d_1 is greedy at $\mathbf{u} = \mathbf{w} - \epsilon\mathbf{1}$ and d_2 is greedy at $\mathbf{v} = \mathbf{w} + \epsilon\mathbf{1}$ for small $\epsilon > 0$. Thus $\|\mathbf{P}_{d_1} - \mathbf{P}_{d_2}\| > 0$ will remain constant even when ϵ becomes extremely small so the bound cannot hold.

Sufficient conditions for (5.195) to hold[60] include that for each $s \in S$:

1. A_s compact and convex,
2. $p(j|s,a)$ is affine in a for each $a \in A_s$, and
3. $r(s,a)$ is strictly convex and twice continuously differentiable in $a \in A_s$.

Bibliographic remarks

Howard [1960], in his doctoral thesis, was the first to describe and analyze the discounted model in considerable generality. However, discounting appeared earlier in inventory models such those described in Arrow et al. [1958]. Blackwell [1962] and

[60] Results that describe the form of $\mathbf{v}$-greedy decision rules are referred to as *selection theorems*.

Blackwell [1965] were seminal papers on the theory of discounted Markov decision process models, with the former being easy to read and most relevant to this book.

The use of value iteration and contraction mappings for discounted models originates with the analysis of stochastic sequential games by Shapley [1953]. However, value iteration has antecedents in earlier unpublished work described in Bellman [1957].

Policy iteration originates with Howard [1960] and Bellman [1957]. The relationship between policy iteration and Newton's method was formalized in Puterman and Brumelle [1979]. The example that policy iteration requires $|S|$ steps is based on an example described by Tsitsiklis in a personal communication. The discussion of strict improvement for policy iterations is motivated by results on pp. 40-41 in Derman [1970].

The discussion of the span follows Section 6.6.1 of Puterman [1994] which is based on the excellent paper Hübner [1977]; its use in Markov decision processes originates with Bat. Theorem 4.1.3 in Kemeny and Snell [1960] develops a precursor to Proposition 5.4.

Bounds and their use to identify sub-optimal actions originate with MacQueen [1966] and MacQueen [1967]. Porteus [2002] provides a good reference to this material.

Hastings [1968] and Kushner and Kleimnan [1971] independently proposed using Gauss-Seidel iteration to improve the performance of value iteration algorithms. The proof that the Gauss-Seidel operator G is a contraction uses concepts in Bertsekas [2012]. The bounds for GS in Section 5.13.2 are new and use several results from Puterman [1994] regarding representing Gauss-Seidel updates as a regular splitting. The concepts of hybrid algorithms and details on asynchronous value iteration may be found in Bertsekas [2012].

Modified policy iteration was independently proposed by Puterman and Shin [1978] and van Nunen [1976]. Its discussion here follows Puterman [1994] and the references therein. Canbolat and Rothblum [2013] establishes convergence of modified policy iteration and bounds for any starting value. Note that this paper originally appeared as an unpublished manuscript in 1979 and was published posthumously in 2013. See also Bertsekas [2012], who refers to this algorithm as *optimistic* policy iteration. At this point in time, we think *approximate* policy iteration would have been a better descriptor.

The linear programming formulation of discounted models originates with D'Epenoux [1960]. Kallenberg [1983] provides a comprehensive study of this model under a range of optimality criteria. Hillier and Lieberman [2021] provides a good introduction to linear programming and Bertsimas and Tsitsiklis [1997] provides a more theoretical treatment.

Our formulation and analysis of the clinical decision making application is based on Alagoz et al. [2007].

Exercises

1. Consider Exercise 1 from Chapter 2. Assume $\lambda = 0.9$.

(a) Write out the Bellman equation for the value function and the state-action value function.

(b) Find an ϵ-optimal policy using value iteration and Gauss-Seidel iteration. Plot the norm and span for each as a function of the iteration number.

(c) Evaluate the bounds for value iteration and Gauss-Seidel iteration after 10, 20 and 30 iterates.

(d) Apply an asynchronous value iteration algorithm in which the state to update is chosen from a uniform distribution on the set of states.

(e) Apply value iteration with action elimination.

(f) Solve the problem with policy iteration and a variant that re-evaluates the value function as soon as a strictly improving action is identified. Compare the convergence and progress of these two variants of policy iteration.

(g) Formulate and solve the problem using linear programming.

2. Find an ϵ-optimal policy for the model in Exercise 1 from Chapter 2 using the following variants of modified policy iteration.

(a) Initiate it with $\mathbf{v} = \mathbf{0}$.

(b) Initiate it with a decision rule of your choice.

(c) Choose m_n fixed, increasing and decreasing.

(d) Integrate the algorithm with Gauss-Seidel updating where appropriate.

(e) Summarize your conclusions on the best approach for implementing modified policy iteration.

3. Consider an infinite horizon version of the periodic inventory review problem of Section 3.1 with a finite warehouse capacity of 10 units, no backlogging, fixed ordering cost of \$12, per unit ordering cost of \$2, revenue of \$5 per item sold, and holding cost of \$1 per unit per period. Assume a scrap value of \$1 per item. If demand cannot be fulfilled by stock on hand after receipt of an order, it is lost. Assume demand is geometrically distributed with $p = 0.3$.

(a) Solve the problem assuming there is no delay between when the order is placed and it arrives. Does the optimal policy have any obvious structure?

(b) Solve the problem assuming that when an order is placed it arrives at the end of the period after demand is met from stock on hand. How do your results compare to those when the stock arrives instantaneously?

4. Consider the replacement model in Section 5.10.2 with $M = 100$, $p = 0.5$, $\lambda = 0.9$ and $K = 50$.

(a) Use policy iteration to find an optimal policy and comment on its structure.

(b) Use linear programming to find an optimal policy.

(c) Develop a policy iteration type algorithm that exploits the result that there exists an optimal control limit policy for this model.

5. Solve a discounted infinite horizon version of the restaurant management problem (Exercise 15 in Chapter 3) using $\lambda = 0.9$.

6. Consider a discounted infinite horizon version of the queuing admission control model of Section 3.4.2. Assume the arrival probability is $b = 0.3$, the service probability is $w = 0.2$, the holding cost is $h(j) = 3j$, the payment is $R = 60$ and the discount rate is 0.95. Moreover, assume that the queue capacity is 100 and that the controller must reject a job if the queue is full.

   (a) Write the Bellman equation for this model.

   (b) Find an optimal policy using policy iteration.

   (c) Find an ϵ-optimal policy using value iteration and modified policy iteration.

   (d) Prove that there exists an optimal control limit policy.

7. **Knowing when to quit.** Consider a game in which transitions occur between states according to a random walk on the integers $\{0, \ldots, M\}$ with transition probabilities $p(j+1|j) = 1 - p(j-1|j) = q$ for $1 \leq j \leq M-1$. Assume further that the game stops when either state 0 or state M is reached. In state 0, the player receives 0 and in state M, the player receives M.

   At each decision epoch the decision maker may quit the game and receive a reward of j if the game is in state $j < M$ or continue playing. Recall that stopping corresponds to a transition to a reward-free absorbing state Δ. Assume a discount rate of $\lambda < 1$.

   (a) Provide the Bellman equation for this model.

   (b) Show that the optimal strategy is to stop immediately if $q \leq 0.5$.

   (c) Find an optimal strategy for all combinations of $M \in \{20, 100\}, q \in \{0.6, 0.75, 0.9\}$ and $\lambda \in \{0.60, 0.90, 0.99\}$ using policy iteration and linear programming.

   (d) Verbally describe the optimal policy and how it varies with the model parameters.

   (e) *Derive the optimal policy analytically.

   (f) How would results change if on stopping in state s, the player receives s^2 or $\sqrt{s}$?

8. Solve the primal and dual linear programs for the two-state model for $\lambda = 0.9$ and $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$. Verify that Strong Duality holds, and that the optimal policies are the same as the ones derived through value iteration and policy iteration.

9. **Discounted optimal stopping.**[61] The purpose of the exercise is to investigate policy evaluation in a simple optimal stopping model. Consider an optimal stopping problem on states $S = \{s_1, s_2, s_3, s_4\}$. Assume in states s_1 and s_2 stopping is possible and in states s_3 and s_4 stopping is not possible. Stopping yields a reward $R = 25$ and continuing yields a reward of 1 in states s_1, s_2, s_3 and 0 in s_4. Let Δ denote the stopped state. Assume the unstopped Markov chain transitions are determined by the transition probability matrix

$$\mathbf{P} = \begin{bmatrix} 0.98 & 0.01 & 0.01 & 0 \\ 0.01 & 0.98 & 0.01 & 0 \\ 0.005 & 0.005 & 0.985 & 0.005 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

   (a) Evaluate the policy d^∞ that stops in state s_1 and continues otherwise. Assume $\lambda = 0.999$.

      i. Evaluate this policy by solving by solving $(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{r}_d$.

      ii. Evaluate this policy using the recursion $\mathbf{v}' = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}$. How many iterations are required to accurately compute $\mathbf{v}$?

      iii. Evaluate this policy using the component form recursion

$$v'(s) = r(s, d(s)) + \lambda \sum_{j \in S} p(j|s, d(s))v(j)$$

      where the summation is computed recursively.

   (b) Find an optimal policy for the optimal stopping problem.

   (c) Investigate the sensitivity of the optimal policy to the value of R.

10. Consider a version of the liver transplantation model of Section 5.12 with $H = 2$ and $L = 2$[62].

   (a) Depict the model symbolically in the form of Figure 2.7 in two ways, one in which the state corresponding to death from transplant failure is distinguished from the state which results from accepting an organ for transplantation, and one in which these two stopped states are combined.

   (b) Using the later specification, provide the transition probability matrix where the two-dimensional state is indexed by (h, l). Also specify the reward function $r((h, l), a)$.

   (c) Using parameters of your choice and the discount rate $\lambda = 0.9997$, evaluate the policy that accepts an organ only in state $(2, 2)$ in three ways:

[61] This example is a good jumping off point for analyzing the transplant model in the next few exercises.

[62] We used this example to debug our code for solving the model in Section 5.12.

i. By solving $(\mathbf{I} - \lambda\mathbf{P})\mathbf{v} = \mathbf{r}$.

ii. Using an iterative scheme in vector form $\mathbf{v}' = \mathbf{r} + \lambda\mathbf{P}\mathbf{v}$. How many iterations are required to accurately compute the exact value found by solving the linear equation?

iii. Using a similar recursion in component form in which the summations are computed recursively.

11. This example revisits the liver transplantation application formulated and analyzed in Section 5.12 using policy iteration and value iteration. Consider reducing the annual discount rate to 0.8. Determine the daily discount rate, generate a policy map, and compare it to Figure 5.19a. Are the changes in the policy map intuitive, given the change in the discount factor?

12. Consider a variant of the liver transplantation problem with only one organ type.

    (a) Solve the model numerically using the specification in Section 5.12 when only an organ of quality 1 is available.

    (b) Repeat your analysis when only an organ of quality 10 is available and compare your results.

    (c) Provide conditions under which the optimal policy is a control limit policy in the health state. Prove that the optimal policy has this form.

    (d) Solve a variant of the problem where there is a single health state (state 6) and multiple organ types. Be sure to redefine transition probabilities.

13. **A really neat problem**[63] Suppose d and e are two decision rules in D^{MD} and define a new decision rule $f \in D^{\text{MD}}$ by

$$f(s) = \begin{cases} d(s) & \text{whenever } v_\lambda^{d^\infty}(s) \geq v_\lambda^{e^\infty}(s) \\ e(s) & \text{whenever } v_\lambda^{d^\infty}(s) < v_\lambda^{e^\infty}(s) \end{cases}$$

    Show that $v_\lambda^{f^\infty}(s) \geq \max\{v_\lambda^{d^\infty}(s), v_\lambda^{e^\infty}(s)\}$ for all $s \in S$.

14. Show that the third term of equation (5.14) in the sum over n is equal to (5.17).

15. Show that $\text{sp}(\mathbf{v})$ is a semi-norm on V.

16. Prove that $\text{sp}(\mathbf{v})$ is a semi-norm and in addition satisfies the three properties following its definition.

17. Evaluate γ and γ' for each of the three other Markovian deterministic policies using the approach in Example 5.7 and equations (5.80) and (5.81). Also, compute the eigenvalues of each of the transition matrices and compare them to γ.

[63] This was on author MLP's final exam when he took a Markov decision process course in graduate school many, many years ago.

18. Evaluate the bounds in Example 5.9 for $\mathbf{v} = (0, 0)$, $\mathbf{v} = (-10, -10)$, and $\mathbf{v} = (20, 20)$. Comment on the tightest of the generated bounds from these value functions.

19. Implement value iteration in Example 5.11 using $\mathbf{v}^0 = (0, 0)$, $\mathbf{v}^0 = (-10, -10)$, and $\mathbf{v}^0 = (20, 20)$ with the span-based stopping criterion. Comment on convergence as a function of the initial value function.

20. Suppose $\pi^* = (d_1, d_2, d_3, \ldots) \in \Pi^{\text{HR}}$ is an optimal policy. Show that $(d_1)^\infty$ is a stationary optimal policy.

21. Prove a variant of Proposition 5.9 when there are strict improvements in two states.

22. Provide a formal proof of Theorem 5.33 using induction and Lemma 5.6.

23. Prove Lemma 5.7.

24. Write out and prove Lemma 5.4 in component notation.

25. Of the four value functions in Example 5.5, verify that $\mathbf{v}_\lambda^{d_4^\infty}$ is the only one that satisfies the Bellman equation and does so for arbitrary λ.

26. Letting $\mathbf{y}_d$ be the vector of dual variables corresponding to the constraints in the vector version of the primal LP, formulate the dual LP in vector notation. Show that by creating a suitable "long" vector, the dual LP can be written as a standard form LP.

27. Prove Theorem 5.25.

28. Consider the preventive maintenance example from Section 5.10.2.

    (a) Verify the result that a control limit policy is optimal by numerically solving the problem over a range of parameter values of your choosing.

    (b) Develop an algorithm for finding an optimal control limit policy by searching only within the class of control limit policies.

    (c) *Provide conditions on $f(s)$ and p that ensure that $0 \leq s^* \leq M - 1$ where s^* is defined in (5.176).

29. Apply Gauss-Seidel iteration directly in Example 5.20 and verify that it gives the same result.

30. Suppose in Example 5.19 that

$$P^\pi[Y_2 = a | X_2 = u, Y_1 = a, X_1 = w] = 0.5$$
$$P^\pi[Y_2 = a | X_2 = u, Y_1 = b, X_1 = w] = 0.3.$$

(a) Compute $w_{d_2'}(a|u,w)$ and $w_{d_2'}(b|u,w)$ where these quantities are defined in (5.184).

(b) What quantities are needed to specify d_2' ?

(c) Suppose the initial state distribution is $\rho(u) = 0.75$ and $\rho(w) = 0.25$. Compute the probability $w_{d_2'}(a|u)$ under this initial state distribution.