Introduction to Markov Decision Processes

Martin L. Puterman and Timothy C. Y. Chan

July 29, 2023

Chapter 1

Infinite Horizon Discounted Models

We welcome all feedback and suggestions at: martin.puterman@sauder.ubc.ca and tcychan@mie.utoronto.ca

"Compound interest is the eighth wonder of the world. He who understands it, earns it . . . , he who doesn't, . . . pays it. "
Albert Einstein, physicist, 1879-1955.

Chapter 1 concerns *infinite horizon Markov decision processes under the expected total discounted reward criterion.* To avoid the above long-winded expression we refer to them 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 **??**, 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 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 $\lambda = 1/(1+i)$. 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 1.1. 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 $1 - \lambda$.

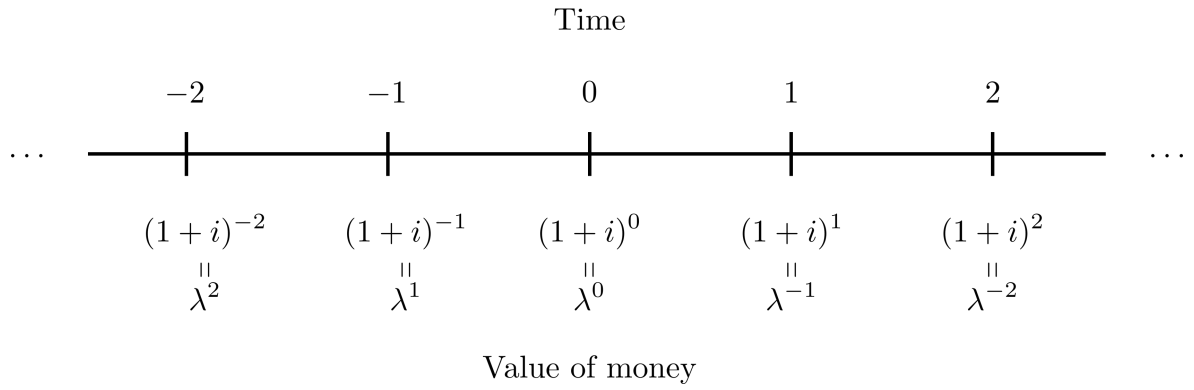


Figure 1.1: Compound interest in the forward time direction is equivalent to discounting in the reverse time direction. The interest rate is i and the discount rate is λ. For example, if you invest \$1 at time 0, you will receive $(1+i)^2$ at time $t = 2$. Conversely, \$1 now is equivalent to λ^2 at time $t = -2$.

This (long) chapter unfolds as follows. It establishes the optimality of stationary deterministic policies in several steps culminating with demonstrating the existence and uniqueness of the solution to the Bellman equation. We then study three classes of iterative algorithms:

- Value iteration
- Policy iteration
- Modified policy iteration

In each case we describe the algorithm and its variants, provide stopping rules, prove convergence and illustrate it through examples. We then discuss the conceptual equivalence between Markov decision processes and linear programming. We revisit several applications with discounted models and show how to extend the results on structured optimal policies in Section **??** to discounted models.

We use vector and matrix notation and introduce appropriate 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.

1.1 Preliminaries

We introduce some notation and basic recursions in this section.

1.1.1 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, such as those considered in the remainder of this book, equations will appear more familiar and more transparent if they are expressed in terms of $r_n(s,a)$ instead of $r_n(s,a,j)$. This may be the case when the reward does not depend on j such as in the queuing control models in Section **??**. 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

$$r_n(s,a) = \sum_{j \in S} r_n(s,a,j) p_n(j|s,a). \tag{1.1}$$

We illustrate this calculation for the two-state model in Example **??** as follows.

Example 1.1. Consider Example **??**. Assume the rewards and probabilities are stationary. Then,

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

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 $r(s_2,a_{2,2}) = 2$.

We emphasize that 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. When we turn to simulation methods in Chapters **??** and **??**, we will retain $r_n(s,a,j)$ as a primitive when appropriate.

1.1.2 Key assumptions

In addition to the standard assumptions we have made in this book about finite states and finite actions, we make the following additional assumptions in 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 $0 \leq \lambda < 1$.

1.1.3 Markovian policies are sufficient

Chapter **??** (Theorem **??**) 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 our development in this chapter, we will start from Markovian (randomized) policies. The following result formally justifies this simplification. A proof of it appears in Appendix 1.12.1 at the end of this chapter. It is the consequence of technical Lemma 1.9, 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 1.1. *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 total discounted reward conditional on $X_1 = s$.*

1.1.4 Matrix and vector representation for stationary policies

To streamline results and simplify many proofs and formulae, we will often represent value functions and rewards by (column) vectors and transition probabilities by matrices. We denote vectors and matrices with **bold** symbols. We provide definitions appropriate when S is finite. All can be extended to S countable, continuous or abstract but such generality will not be needed here.

Matrix and vector notation

V: A vector or linear space[a] of real-valued $|S|$-dimensional vectors.
$\mathbf{v}$: An $|S|$-dimensional vector usually representing a value function.
$\mathbf{r}_d$: An $|S|$-dimensional vector of rewards corresponding to $d \in D^{\text{MR}}$ (see equation (1.2)).
$\mathbf{e}$: A vector with all components equal to one.
$\mathbf{e}_s$: A vector with a 1 in the s-th component and zeros elsewhere.
$\mathbf{0}$: A vector with all components equal to zero.
$\mathbf{P}_d$: A $|S| \times |S|$-dimensional transition probability matrix corresponding to $d \in D^{\text{MR}}$ (see equation (1.3)).
$\mathbf{I}$: A $|S| \times |S|$-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 $V \to V$.

We adopt the following further notational conventions. 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, $(v(s_1), \ldots, v(s_M))$ 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 we multiply a vector $\mathbf{v}$ by a matrix $\mathbf{P}_d$, we write $\mathbf{P}_d\mathbf{v}(s_k)$ for the s_k-th component of the resulting vector. Notice that the vector $\mathbf{v}$ remains bold here, since we are not taking 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 non-linear) 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 we write functions as $f(x)$, we do not use parentheses when writing $T\mathbf{v}$.[1] We write $T\mathbf{v}(s)$ to indicate the s-th component of $T\mathbf{v}$ and note that $T^n\mathbf{v} = T \cdots T\mathbf{v}$.

[1]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, we write $\mathbf{A}\mathbf{v}$ instead of $\mathbf{A}(\mathbf{v})$.

Rewards and transition probabilities corresponding to a Markovian policies.

We now show how to construct $\mathbf{r}_d$ 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

$$r_d(s) = \begin{cases} r(s, d(s)) & d \in D^{\mathrm{MD}} \\ \sum_{a \in A_s} w_d(s,a) r(s,a) & d \in D^{\mathrm{MR}}. \end{cases} \tag{1.2}$$

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

$$P_d(j|s) = \begin{cases} p(j|s, d(s)) & d \in D^{\mathrm{MD}} \\ \sum_{a \in A_s} w_d(s,a) p(j|s,a) & d \in D^{\mathrm{MR}}. \end{cases} \tag{1.3}$$

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

$$\mathbf{P}_\pi^n := \mathbf{P}_{d_1} \mathbf{P}_{d_2} \cdots \mathbf{P}_{d_n} \tag{1.4}$$

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 $\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} \in V$.

1.1.5 Partial orders

To express the Bellman equation in vector notation, we need to explain what we mean by a "max" over a set of vectors. Since the definition of a vector space does not include any notion of an ordering of vectors, we introduce the concept of a *partial order*. For our purposes, we use 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.

Since we consider the component-wise partial order, we need to define a corresponding component-wise maximum.

Definition 1.1. 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 $v^*(s) = v_k(s)$ for at least one $k = 1, \ldots, K$ and some $s \in S$. We use c-max to denote the component-wise maximum, written as

$$\mathbf{v}^* := \text{c-max}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K\} \tag{1.5}$$

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

$$k^* \in \underset{k=1,\ldots,K}{\arg\text{c-max}}\{\mathbf{v}_k\} \tag{1.6}$$

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 1.2. 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 $\mathbf{v}^* \notin \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ in this case.

Where we use the component-wise maximum in a Markov decision process model, the situation in Example 1.2 does not occur. In particular, we will use c-max when taking the maximum over deterministic decision rules, where each decision rule is a vector 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 1.3. 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.

We emphasize that the 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 we show later.

1.1.6 Norms

To establish the convergence of a sequence of iterates of a computational algorithm, we will require a notion of distance in a vector space. To do this we define a *norm* on V.

Definition 1.2. For $\mathbf{v} \in V$, define the *norm* of $\mathbf{v}$ by

$$\|\mathbf{v}\| := \max_{s \in S} |v(s)|. \tag{1.7}$$

This norm is often called the *sup-norm* norm. 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 (1.7) by a supremum (hence the name sup-norm). However, we will not require that degree of generality 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 we will use only the sup-norm[2].

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

Definition 1.3. For $\mathbf{Q} \in V \times V$, define the *matrix norm* of $\mathbf{Q}$ by

$$\|\mathbf{Q}\| := \sup_{\|\mathbf{v}\|=1} \|\mathbf{Q}\mathbf{v}\| = \max_{s \in S} \sum_{j \in S} |q(s, j)|. \tag{1.8}$$

Note that we sometimes refer to matrices as *linear operators* to distinguish them from more general non-linear operators.

The following is a useful inequality.

Lemma 1.1. For $\mathbf{v} \in V$ and matrix $\mathbf{Q} : V \to V$,

$$\|\mathbf{Q}\mathbf{v}\| \leq \|\mathbf{Q}\|\|\mathbf{v}\|. \tag{1.9}$$

Combining the concepts of partial order and norm with a vector space, we have the following definition.

Definition 1.4. A vector space V together with a partial order $\geq$ and a norm $\|\cdot\|$ is referred to as a *partially ordered normed linear space.*

In this book (except Chapter **??**) we will restrict attention to the vector space V of real-valued $|S|$-dimensional vectors with component-wise partial order and sup-norm.

1.2 The Expected Total Discounted Reward of a Policy

Define $v_\lambda^\pi(\mathrm{s})$, the expected total discounted reward of policy $\pi \in \Pi^{\mathrm{HD}}$ starting in state s with discount factor λ, by[3]

[2]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, we will drop the subscript on the infinity norm.

[3]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

$$\boxed{v_\lambda^\pi(s) := \lim_{N\to\infty} E^\pi\left[\sum_{n=1}^{N} \lambda^{n-1} r(X_n, Y_n) \,\middle|\, X_1 = s\right]. \qquad (1.10)}$$

We now obtain two equivalent representations for $v_\lambda^\pi(s)$. Be sure to note the subtle distinctions between them. Since we assume bounded rewards and discount factor $0 \le \lambda < 1$, the bounded convergence theorem[4] justifies interchanging the limit and the expectation above to obtain

$$v_\lambda^\pi(s) = \lim_{N\to\infty} E^\pi\left[\sum_{n=1}^{N} \lambda^{n-1} r(X_n, Y_n) \,\middle|\, X_1 = s\right] = E^\pi\left[\sum_{n=1}^{\infty} \lambda^{n-1} r(X_n, Y_n) \,\middle|\, X_1 = s\right]. \tag{1.11}$$

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

$$v_\lambda^\pi(s) = \lim_{N\to\infty} \sum_{n=1}^{N} \lambda^{n-1} E^\pi\left[r(X_n, Y_n) \,\middle|\, X_1 = s\right] = \sum_{n=1}^{\infty} \lambda^{n-1} E^\pi\left[r(X_n, Y_n) \,\middle|\, X_1 = s\right]. \tag{1.12}$$

This representation on the right hand side of (1.12) will provide the basis for most of the following development.

1.2.1 Insights into how the expected discounted reward is evaluated

This section shows why we consider the second representation for $v_\lambda^\pi(s)$ so important, by looking into the expectations on the right hand side of (1.12) in depth. For any n,

$$E^\pi[r(X_n, Y_n)|X_1 = s] = \sum_{j\in S} \sum_{a\in A_j} r(j,a) p^\pi(X_n = j, Y_n = a | X_1 = s), \tag{1.13}$$

where p^π is a probability distribution incorporating both the transition probabilities and the action randomization distribution (see Section **??**). Thus combining (1.12) and (1.13),

$$v_\lambda^\pi(s) = \sum_{n=1}^{\infty} \sum_{j\in S} \sum_{a\in A_j} \lambda^{n-1} r(j,a) p^\pi(X_n = j, Y_n = a | X_1 = s). \tag{1.14}$$

distributed according to $w_d(X_n)$.

[4]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

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

Because S is finite in our models, we replace the integrals by sums.

As a consequence of Theorem 1.1, we restrict our focus to $\pi = (d_1, d_2, \ldots) \in \Pi^{\text{MR}}$ without loss of generality.

To gain insight into equation (1.14), we explicitly write out the first three terms in the summation over n as follows:

$$n = 1: \quad \sum_{a \in A_j} r(s,a) w_{d_1}(s,a) \tag{1.15}$$

$$n = 2: \quad \lambda \sum_{j \in S} \sum_{a' \in A_j} r(j,a') w_{d_2}(j,a') \sum_{a \in A_s} p(j|s,a) w_{d_1}(s,a) \tag{1.16}$$

$$n = 3: \quad \lambda^2 \sum_{k \in S} \sum_{a'' \in A_j} r(k,a'') w_{d_3}(k,a'') \sum_{j \in S} \sum_{a' \in A_j} p(j|j,a') w_{d_2}(j,a') \sum_{a \in A_s} p(j|s,a) w_{d_1}(s,a). \tag{1.17}$$

To derive the first term, note that states $j \neq s$ will have probability zero and $p^\pi(X_1 = s, Y_1 = a | X_1 = s) = p^\pi(Y_1 = a | X_1 = s) = w_{d_1}(s,a)$ 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 the state is s.

Using the law of total probabilities and the Markov property, the second term follows by

$$\begin{aligned}
p^\pi(X_2 = j, Y_2 = a' | X_1 = s) &= p^\pi(Y_2 = a' | X_2 = j, X_1 = s) p^\pi(X_2 = j | X_1 = s) && (1.18)\\
&= w_{d_2}(j,a') p^\pi(X_2 = j | X_1 = s) && (1.19)\\
&= w_{d_2}(j,a') \sum_{a \in A_s} p^\pi(X_2 = j | X_1 = s, Y_1 = a) p^\pi(Y_1 = a | X_1 = s) && (1.20)\\
&= w_{d_2}(j,a') \sum_{a \in A_s} p(j|s,a) w_{d_1}(s,a). && (1.21)
\end{aligned}$$

Thus (1.16) represents the discounted expected 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 (1.17) 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 the equation (1.10) (or equivalently (1.14)), we briefly discuss how to approximate it for a given policy $\pi = (d_1, d_2, \ldots)$ using Monte Carlo simulation. More details on how to implement it and other approaches appear in Chapter **??**.

A challenge when simulating discounted rewards is to account for the infinite number of terms in (1.14). This can be achieved through truncation or using geometric stopping times. Putting that issue aside until Chapter **??**, we assume that we seek to obtain an estimate of the expected discounted reward after truncating it to K terms. We express the algorithm in terms of $r(s, a, j)$ and assume we know $p(j|s, a)$.

Simulating one replication of an estimate of $v_\lambda^\pi(s)$ for fixed $s \in S$.

```
1. Initialize: Specify $\pi = (d_1, d_2, \ldots) \in \Pi^{\mathrm{MR}}$, $k \leftarrow 1$, sum $\leftarrow 0$. Choose $s \in S$.
2. While $k < K$:
    (a) Sample action $a$ from $w_{d_k}(s, \cdot)$
    (b) Sample $s'$ from $p(\cdot|s, a)$
    (c) sum $\leftarrow$ sum + $\lambda^{k-1} r(s, a, s')$
    (d) $s \leftarrow s'$
    (e) $k \leftarrow k + 1$
```

Note that if the reward was not a function of the next state we could interchange 2(b) and 2(c) and replace the former by sum $\leftarrow$ sum + $\lambda^{k-1} r(s, a)$. By generating many sample paths in this way, we can obtain an estimate of the *distribution* of $v_\lambda^\pi(s)$.

1.2.2 Representing value functions in vector form

Now, we show how to express quantities in the previous section using vector notation. We rewrite the first three terms in the sum over n in equation (1.14) as

$$n = 1: \quad r_{d_1}(s) \tag{1.22}$$

$$n = 2: \quad \lambda \sum_{j \in S} r_{d_2}(j) P_{d_1}(j|s) \tag{1.23}$$

$$n = 3: \quad \lambda^2 \sum_{j \in S} r_{d_3}(j) P_{d_2}(j|k) P_{d_1}(k|s). \tag{1.24}$$

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 as $\lambda^2 \mathbf{P}_\pi^2 \mathbf{r}_{d_3}$ using the 2-step transition probability matrix $\mathbf{P}_\pi^2$.

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

$$\boxed{\mathbf{v}_\lambda^\pi = \sum_{n=1}^{\infty} \lambda^{n-1} \mathbf{P}_\pi^{n-1} \mathbf{r}_{d_n} \qquad (1.25)}$$

This infinite sum converges because

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

Expanding this sum and grouping terms appropriately, we obtain

$$\mathbf{v}_\lambda^\pi = \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \mathbf{r}_{d_2} + \lambda^2 \mathbf{P}_{d_1} \mathbf{P}_{d_2} \mathbf{r}_{d_3} + \cdots \qquad (1.26)$$

$$= \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \left(\mathbf{r}_{d_2} + \lambda \mathbf{P}_{d_2} \mathbf{r}_{d_3} + \cdots \right). \qquad (1.27)$$

Thus, if we define a new policy $\pi' = (d_2, d_3, \ldots)$ that consists of all the decision rules from d_2 onwards, we can write

$$\mathbf{v}_\lambda^\pi = \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \mathbf{v}_\lambda^{\pi'}. \qquad (1.28)$$

This equation can be interpreted as follows: the expected total 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 total discounted reward over the infinite horizon, following policy π' from the second decision epoch onwards. Equivalently, we can interpret this equation as an expression for the expected value in a one-period problem, where the terminal reward is the expected total discounted reward associated with π'.

In component form, equation (1.28) can be written as

$$v_\lambda^\pi(s) = r_{d_1}(s) + \lambda \sum_{j \in S} P_{d_1}(j|s) v_\lambda^{\pi'}(j), \qquad (1.29)$$

for all $s \in S$. If $\pi \in \Pi^{\text{MD}}$, then equation (1.28) becomes

$$v_\lambda^\pi(s) = r(s, d_1(s)) + \lambda \sum_{j \in S} p(j|s, d_1(s)) v_\lambda^{\pi'}(j) \qquad (1.30)$$

for all $s \in S$. The subtle difference between (1.29) and (1.30) is that the former accommodates randomized decision rules (see equations (1.2) and (1.3)). Notice that equation (1.30) is the discounted, infinite horizon generalization of equation (**??**) from the finite horizon setting with stationary rewards and transition probabilities.

Writing the above two expressions as expectations will be especially useful in Chapter **??**. Regardless of whether π is deterministic or randomized, we can write

$$v_\lambda^\pi(s) = E^\pi[r(X_1, Y_1) + \lambda v_\lambda^{\pi'}(X_2) | X_1 = s]. \qquad (1.31)$$

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

$$v_\lambda^\pi(s) = E^\pi[r(X_1, Y_1, X_2) + \lambda v_\lambda^{\pi'}(X_2) | X_1 = s]. \qquad (1.32)$$

1.2.3 Evaluating stationary policies

Stationary policies play an especially important role in discounted models. We now show how restricting attention to such policies greatly simplifies the above expressions and more. When π is stationary, then $\pi = \pi'$ in (1.28). Let $d^\infty := (d, d, \ldots)$ denote the stationary policy that uses decision rule $d \in D^{\text{MR}}$ in every epoch. Then equation (1.25) becomes

$$\mathbf{v}_\lambda^{d^\infty} = \sum_{n=0}^{\infty} \lambda^n \mathbf{P}_d^n \mathbf{r}_d \tag{1.33}$$

and equation (1.28) becomes

$$\mathbf{v}_\lambda^{d^\infty} = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}_\lambda^{d^\infty}. \tag{1.34}$$

In other words, $\mathbf{v}_\lambda^{d^\infty}$ satisfies the system of linear equations:

$$\mathbf{v} = \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} \tag{1.35}$$

In fact, a key result shows that $\mathbf{v}_\lambda^{d^\infty}$ is the unique solution of equation (1.35) and has a succinct algebraic representation. It is a fundamental result that we will use extensively in the remainder of this chapter. Before we prove this result, we state the following lemma, whose proof is given in the Appendix. It provides the matrix generalization of the representation for a convergent geometric series $\sum_{n=0}^{\infty} a^n = (1 - a)^{-1}$ when $|a| < 1$.

Lemma 1.2. 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

$$(\mathbf{I} - \mathbf{Q})^{-1} = \sum_{n=0}^{\infty} \mathbf{Q}^n. \tag{1.36}$$

Theorem 1.2. Suppose $0 \leq \lambda < 1$ and $d \in D^{\text{MR}}$. Then

$$\mathbf{v}_\lambda^{d^\infty} = (\mathbf{I} - \lambda \mathbf{P}_d)^{-1} \mathbf{r}_d \tag{1.37}$$

and it is the unique solution of equation (1.35).

Proof. Let $d \in D^{\text{MR}}$. Rewriting (1.35), we have

$$(\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} = \mathbf{r}_d. \tag{1.38}$$

What remains is to show that the matrix $\mathbf{I} - \lambda \mathbf{P}_d$ is invertible, which follows from Lemma 1.2 since each component of $\lambda \mathbf{P}_d$ is strictly less than 1[5] □

The following example illustrates the computation of $\mathbf{v}_\lambda^{d^\infty}$ in two ways: direct solution of the system of linear equations (1.35) and using representation (1.37).

Example 1.4. Consider a discounted, infinite horizon version of Example **??** with a randomized stationary policy d^∞ that, in every epoch, chooses action $a_{1,1}$ in state s_1 and chooses state $a_{2,1}$ and $a_{2,2}$ with equal probability in state s_2. First, we provide a direct solution of equation (1.35).

In component form,

$$v(s_1) = 5 \cdot 0.8 - 5 \cdot 0.2 + \lambda(0.8v(s_1) + 0.2v(s_2)) \tag{1.39}$$

$$\begin{aligned} v(s_2) = &\ 0.5(-5 \cdot 1) + 0.5(20 \cdot 0.4 - 10 \cdot 0.6) \\ &+ \lambda((0.5 \cdot 0 + 0.5 \cdot 0.4)v(s_1) + (0.5 \cdot 1 + 0.5 \cdot 0.6)v(s_2)), \end{aligned} \tag{1.40}$$

which simplifies to

$$v(s_1) = 3 + \lambda(0.8v(s_1) + 0.2v(s_2)) \tag{1.41}$$

$$v(s_2) = -1.5 + \lambda(0.2v(s_1) + 0.8v(s_2)). \tag{1.42}$$

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

$$v_\lambda^{d^\infty}(s_1) = \frac{3(1-2.7\lambda)}{(1-\lambda)(1-0.6\lambda)}, \qquad v_\lambda^{d^\infty}(s_2) = \frac{3(0.6\lambda - 0.5)}{(1-\lambda)(1-0.6\lambda)}. \tag{1.43}$$

For comparison, let's compute $\mathbf{v}_\lambda^{d^\infty}$ using equation (1.37).

$$\mathbf{I} - \lambda \mathbf{P}_d = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \lambda \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} = \begin{pmatrix} 1-0.8\lambda & -0.2\lambda \\ -0.2\lambda & 1-0.8\lambda \end{pmatrix} \tag{1.44}$$

Then

$$(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}\mathbf{r}_d = \frac{1}{(1-\lambda)(1-0.6\lambda)} \begin{pmatrix} 1-0.8\lambda & 0.2\lambda \\ 0.2\lambda & 1-0.8\lambda \end{pmatrix} \begin{pmatrix} 3 \\ -1.5 \end{pmatrix}, \tag{1.45}$$

which can easily be verified to be the same as (1.43).

[5]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 **??** 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.

We now compute the expected discounted reward for the four deterministic stationary policies using (1.37). 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 $d_4 = (a_{1,2}, a_{2,2})$, we find that

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

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. We show below that this implies d_4^∞ is optimal.

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

Lemma 1.2 provides the basis for an important series expansion for $(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}$.

Theorem 1.3. Suppose $0 \leq \lambda < 1$ and $d \in D^{\text{MR}}$, then

$$(\mathbf{I} - \lambda \mathbf{P}_d)^{-1} = \sum_{n=0}^{\infty} \lambda^n \mathbf{P}_d^n. \tag{1.46}$$

The following lemma summarizes several properties of $(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}$ that we will use throughout this chapter.

Lemma 1.3. Suppose $0 \leq \lambda < 1$ and $\mathbf{u}, \mathbf{v} \in V$. Then, for any $d \in D^{\text{MR}}$:

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 1.3,

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

since λ and $\mathbf{P}_d$ are non-negative. The second result follows by applying the first result to $\mathbf{u} - \mathbf{v}$. The third result follows from (1.46), the fact that $\mathbf{P}_d \mathbf{e} = \mathbf{e}$ and the formula for the sum of a geometric series. □

1.3 Optimal policies and the Bellman Equation

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

1.3.1 Optimal policies

We begin by formally defining optimal policies and optimal value functions for infinite horizon discounted models.

Definition 1.5. An *optimal policy* $\pi^* \in \Pi^{\mathrm{HR}}$ satisfies

$$v_\lambda^{\pi^*}(s) \geq v_\lambda^{\pi}(s) \tag{1.47}$$

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

Definition 1.6. The *optimal value function* $v_\lambda^*(s)$ is defined by

$$v_\lambda^*(s) := \sup_{\pi \in \Pi^{\mathrm{HR}}} v_\lambda^{\pi}(s) \tag{1.48}$$

for all $s \in S$.

Definition 1.7. For any $\epsilon > 0$, an ϵ-optimal policy $\pi^\epsilon \in \Pi^{\mathrm{HR}}$ satisfies

$$v_\lambda^{\pi^\epsilon}(s) \geq v_\lambda^*(s) - \epsilon \tag{1.49}$$

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.** The following theorem summarizes the main result of this section. The rest of the section develops the necessary results to establish Theorem 1.4.

Theorem 1.4. Suppose A_s is finite for each $s \in S$. Then there exists $\pi^* \in \Pi^{\mathrm{SD}}$ 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.

1.3.2 The Bellman equation for a discounted model

In Chapter **??** we showed in the finite horizon model that the following sequence of recursive equations (re-expressed in the notation of this chapter) characterized optimal value functions and policies:

$$v_n(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_{n+1}(j)) \right\}. \tag{1.50}$$

Informally, passing to the limit on both sides of (1.50) suggests that the following expression is the appropriate form of the Bellman equation for a discounted model.

$$v(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \right\}. \tag{1.51}$$

We will show that this equation has a unique solution and that the solution equals the optimal value function.

Vector form of the Bellman equation

In vector notation, we write equation (1.51) as

$$\mathbf{v} = \underset{d \in D^{\mathrm{MD}}}{\text{c-max}} \{ \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} \}. \tag{1.52}$$

It is important to note that the vector Bellman equation is written with c-max instead of max.[6] The use of the expression c-max emphasizes that the maximum of $\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}$ corresponds to a decision rule with components $d^*(s)$, each of which maximizes the expression

$$r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \tag{1.53}$$

over $a \in A_s$ for each $s \in S$. Example 1.2 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. We write $d^* \in \arg\text{c-max}_{d \in D^{\mathrm{MD}}} \{ \mathbf{r}_d + \lambda \mathbf{P} \mathbf{v} \}$ to emphasize that d^* is the component-wise maximum.

[6]The c-max notation is not standard, but in our experience through teaching, using max was a point of confusion that this new nomenclature hopefully resolves. When the equation is written with a max, it is frequently interpreted in the following (incorrect) way, which is contrary to the dynamic programming principle of reducing a complex problem into a series of simpler problems:

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

1.3.3 The maximum return operator

To simplify notation, simplify several proofs below and suggest generalizations to more complex models, we define the non-linear operator $L : V \to V$ by

$$L\mathbf{v} := \underset{d \in D^{\text{MD}}}{\text{c-max}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}. \tag{1.54}$$

The operator L applied to the vector $\mathbf{v}$ returns the vector of values associated with the deterministic decision rule that maximizes the expected total reward for each state s in a one period problem with terminal reward $\mathbf{v}$. We refer to L as the *maximum return operator*. We remind the reader that for $\mathbf{v} \in V$, $L\mathbf{v}$ is a vector in V and its s-th component is represented by $L\mathbf{v}(s)$.

The following lemma is a direct application of Lemma **??** expressed in vector notation. It states that the component-wise maximum over Markovian deterministic decision rules equals the component-wise maximum over Markovian randomized decision rules.

Lemma 1.4. For any $\mathbf{v} \in V$,

$$\underset{d \in D^{\text{MD}}}{\text{c-max}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\} = \underset{d \in D^{\text{MR}}}{\text{c-max}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}. \tag{1.55}$$

Using this notation, the Bellman equation (1.52) can be expressed as

$$\mathbf{v} = L\mathbf{v}. \tag{1.56}$$

A solution of equation (1.56) is called a *fixed point* of L. We will use theoretical results regarding existence of fixed points for contraction mappings, which we define below, to prove that the Bellman equation has a unique solution. For convenience, we also define for $d \in D^{\text{MR}}$ the operator $L_d : V \to V$ by

$$L_d\mathbf{v} := \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}. \tag{1.57}$$

That is, L_d gives the value of using decision rule d for one period and then receiving terminal reward $\mathbf{v}$. Note that we have already seen the fixed point equation $L_d\mathbf{v} = \mathbf{v}$ in (1.35) and shown that $\mathbf{v}_\lambda^{d^\infty}$ is a fixed point of L_d.

Contraction mappings

We begin with a definition of an important analytic concept, which is fundamental to establishing many key results in this chapter as well as in Chapter **??**.

Definition 1.8. We say that an operator $T : V \to V$ is a *contraction mapping* on V with *modulus* K, if for all $\mathbf{u}$ and $\mathbf{v}$ in V there exists a constant $0 \leq K < 1$ for which

$$\|T\mathbf{v} - T\mathbf{u}\| \leq K\|\mathbf{v} - \mathbf{u}\|. \tag{1.58}$$

As a consequence of this definition, for any $n \geq 1$,

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

Contraction mappings are especially important for us when they operate on a complete normed linear space. 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 $\|\mathbf{v}^n - \mathbf{v}^*\| \to 0$.[7] Note that the set of real numbers is complete, while the set of rational numbers is not. In our setting, the set of all $|S|$-dimensional real vectors, V, is complete.

Theorem 1.5. (Contraction Mapping (Banach) Fixed-Point Theorem)
Let V be a complete normed linear space and $T : V \to V$ be a contraction mapping.

1. There exists a unique $\mathbf{v}^* \in V$ such that $T\mathbf{v}^* = \mathbf{v}^*$;
2. For any $\mathbf{v}^0 \in V$, the sequence $\mathbf{v}^n, n = 1, 2, \ldots$, defined by

$$\mathbf{v}^{n+1} = T\mathbf{v}^n = T^{n+1}\mathbf{v}^0 \tag{1.59}$$

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

This result is known as the *Banach Fixed Point Theorem.* Its proof is widely available and surprisingly straightforward. The challenge in greater generality is establishing that V is complete. This result is important because when we establish that L is a contraction mapping, then the first part of Theorem 1.5 establishes the existence of a unique fixed point for L in V. The second part ensures that repeated application of the recursion $\mathbf{v} \leftarrow T\mathbf{v}$, converges to a fixed point of T.

Proposition 1.1. Suppose $0 \leq \lambda < 1$. Then the operator L defined in (1.54) is a contraction mapping on V with modulus λ.

Proof. We prove this result by reverting to component notation. Let $\mathbf{u}$ and $\mathbf{v}$ be elements of V. Fix $s \in S$ and assume without loss of generality that $L\mathbf{v}(s) \geq L\mathbf{u}(s)$.

[7] 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.

Then by the finiteness of A_s, there exists an $a_s^* \in A_s$ for which

$$r(s, a_s^*) + \lambda \sum_{j \in S} p(j|s, a_s^*)v(j) = \max_{a \in A_s}\{r(s, a) + \lambda \sum_{j \in S} p(j|s, a)v(j)\} = L\mathbf{v}(s). \tag{1.60}$$

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

$$r(s, a_s^*) + \lambda \sum_{j \in S} p(j|s, a_s^*)u(j) \leq \max_{a \in A_s}\left\{r(s, a) + \lambda \sum_{j \in S} p(j|s, a)u(j)\right\} = L\mathbf{u}(s). \tag{1.61}$$

Subtracting (1.61) from (1.60) and noting the assumption that $L\mathbf{v}(s) \geq L\mathbf{u}(s)$, it follows that

$$\begin{aligned}
0 &\leq L\mathbf{v}(s) - L\mathbf{u}(s) && (1.62)\\
&\leq r(s, a_s^*) + \lambda \sum_{j \in S} p(j|s, a_s^*)v(j) - \left(r(s, a_s^*) + \lambda \sum_{j \in S} p(j|s, a_s^*)u(j)\right) && (1.63)\\
&\leq \lambda \sum_{j \in S} p(j|s, a_s^*)\|\mathbf{v} - \mathbf{u}\| && (1.64)\\
&= \lambda\|\mathbf{v} - \mathbf{u}\|. && (1.65)
\end{aligned}$$

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

$$\|L\mathbf{v} - L\mathbf{u}\| \leq \lambda\|\mathbf{v} - \mathbf{u}\| \tag{1.66}$$

completing the proof. □

Combining the above results and noting that V is complete leads to the following important result.

Theorem 1.6. Suppose $0 \leq \lambda < 1$. Then $L\mathbf{v} = \mathbf{v}$ has a unique solution in V.

This theorem provides an alternative way of establishing that $\mathbf{v}_\lambda^{d^\infty}$ is the unique fixed point of $L_d\mathbf{v} = \mathbf{v}$ in V, which we previously showed in Theorem 1.2.

Corollary 1.1. 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.

1.3.4 Existence of an optimal value function

That the solution of the Bellman equation is the optimal value function follows from the following important theorem that provides upper and lower bounds on the optimal value function.

Theorem 1.7. Let $0 \le \lambda < 1$. If there exists a $\mathbf{v} \in V$ for which:

1. $\mathbf{v} \ge L\mathbf{v}$, then $\mathbf{v} \ge \mathbf{v}_\lambda^*$,
2. $\mathbf{v} \le L\mathbf{v}$, then $\mathbf{v} \le \mathbf{v}_\lambda^*$ an
3. $\mathbf{v} = L\mathbf{v}$, then $\mathbf{v} = \mathbf{v}_\lambda^*$. Moreover, it is the unique solution of this system of equations.

Proof. To prove the first result, consider an arbitrary policy $\pi = (d_1, d_2, \ldots) \in \Pi^{\mathrm{MR}}$ and $\epsilon > 0$. We will show that $\mathbf{v} \ge \mathbf{v}_\lambda^\pi - \epsilon\mathbf{e}$, from which it follows that

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

where the first equality follows from Theorem 1.1.

Combining $L\mathbf{v} \ge \mathbf{v}$ with Lemma 1.4, it follows that

$$\mathbf{v} \ge \operatorname*{c\text{-}max}_{d \in D^{\mathrm{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\} = \operatorname*{c\text{-}max}_{d \in D^{\mathrm{MR}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\} \ge \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}$$

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

$$\begin{aligned} \mathbf{v} &\ge \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \mathbf{v} && (1.67) \\ &\ge \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1}(\mathbf{r}_{d_2} + \lambda \mathbf{P}_{d_2} \mathbf{v}) && (1.68) \\ &= \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \mathbf{r}_{d_2} + \lambda^2 \mathbf{P}_{d_1} \mathbf{P}_{d_2} \mathbf{v} && (1.69) \end{aligned}$$

Repeating this argument, we have for $n \ge 1$,

$$\begin{aligned} \mathbf{v} &\ge \mathbf{r}_{d_1} + \lambda \mathbf{P}_{d_1} \mathbf{r}_{d_2} + \cdots + \lambda^{n-1} \mathbf{P}_\pi^{n-1} \mathbf{r}_{d_n} + \lambda^n \mathbf{P}_\pi^n \mathbf{v} && (1.70) \\ &= \mathbf{v}_\lambda^\pi + \lambda^n \mathbf{P}_\pi^n \mathbf{v} - \sum_{k=n}^{\infty} \lambda^k \mathbf{P}_\pi^k \mathbf{r}_{d_{k+1}}, && (1.71) \end{aligned}$$

where the last equality follows from the definition of $\mathbf{v}_\lambda^\pi$ in (1.25). For n sufficiently large, each of the last two terms are bounded above in norm by $\epsilon/2$, and the result follows.

To prove the second result we have $\mathbf{v} \le \text{c-max}_{d \in D^{\mathrm{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}$, which implies there exists a specific $d \in D^{\mathrm{MD}}$ such that $\mathbf{v} \le \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}$. Then,

$$\begin{aligned} \mathbf{v} &\le \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} && (1.72) \\ &\le \mathbf{r}_d + \lambda \mathbf{P}_d (\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}) && (1.73) \\ &= \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{r}_d + \lambda^2 \mathbf{P}_d^2 \mathbf{v} && (1.74) \end{aligned}$$

Repeating this argument, we have

$$\begin{aligned}\mathbf{v} &\le \sum_{n=1}^{\infty} \lambda^{n-1}\mathbf{P}_d^{n-1}\mathbf{r}_d && (1.75)\\ &= \mathbf{v}_\lambda^{d^\infty} && (1.76)\\ &\le \sup_{\pi\in\Pi^{\mathrm{HR}}} \mathbf{v}_\lambda^{\pi}, && (1.77)\end{aligned}$$

as desired.

If $\mathbf{v} = L\mathbf{v}$, then $\mathbf{v} = \mathbf{v}_\lambda^*$ and uniqueness follows from the first two results. □

Combining Theorem 1.7 with Theorem 1.6 yields the following fundamental result.

Theorem 1.8. The optimal value function $\mathbf{v}_\lambda^*$ is the unique solution to the optimality equation

$$\mathbf{v} = L\mathbf{v} = \underset{d\in D^{\mathrm{MD}}}{\text{c-max}}\{\mathbf{r}_d + \lambda\mathbf{P}_d\mathbf{v}\}. \quad (1.78)$$

Observe that if we restrict the set of Markovian deterministic decision rules D^{MD} to be the singleton set $\{d\}$ in Theorem 1.8, then we recover Theorem 1.2 and Corollary 1.1, which states that $\mathbf{v}_\lambda^{d^\infty}$ is the unique solution to $\mathbf{v} = \mathbf{r}_d + \lambda\mathbf{P}_d\mathbf{v}$. This observation should also hint at the main result of the next subsection, that if d achieves the maximum in Theorem 1.8, then $\mathbf{v}_\lambda^{d^\infty}$ is the unique optimal solution to the Bellman equation.

We restate the above results in component notation for completeness. Results in this format will be useful in the linear programming section below.

Corollary 1.2. Let $\mathbf{v} \in V$.

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

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

3. The optimal value function $v^*_\lambda(s)$ is the unique solution of

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

1.3.5 Existence of an optimal stationary policy

Now, we establish that a specific stationary policy is optimal. Let

$$d^* \in \arg \operatorname*{c-max}_{d \in D^{\mathrm{MD}}} \{ \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}^*_\lambda \}. \tag{1.79}$$

This means that

$$L\mathbf{v}^*_\lambda = L_{d^*}\mathbf{v}^*_\lambda = \mathbf{r}_{d^*} + \lambda \mathbf{P}_{d^*}\mathbf{v}^*_\lambda = \mathbf{v}^*_\lambda.$$

But $\mathbf{v}^{(d^*)^\infty}_\lambda$ is the unique solution of $L_{d^*}\mathbf{v} = \mathbf{v}$, so $\mathbf{v}^{(d^*)^\infty}_\lambda = \mathbf{v}^*_\lambda$. Hence we have established the following fundamental result.

Theorem 1.9. Let A_s be finite for each $s \in S$ and let

$$d^* \in \arg \operatorname*{c-max}_{d \in D^{\mathrm{MD}}} \{ \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}^*_\lambda \}. \tag{1.80}$$

Then $(d^*)^\infty$ is an optimal deterministic stationary policy.

We restate this result in component notation as follows. For all $s \in S$ define a^*_s by

$$a^*_s \in \arg\max_{a \in A_s} \left\{ r(s,a) + \sum_{j \in S} \lambda p(j|s,a) v^*_\lambda(j) \right\} \tag{1.81}$$

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

Observe that when the arg c-max in some state is not unique, there will be more than one optimal deterministic stationary policy. Moreover any stationary policy that randomizes over actions satisfying (1.81) in state s will also be optimal. This is particularly relevant because some policies with the same expected total discounted reward might have smaller variance.

Thus, to find an optimal policy in a discounted model

1. Solve the Bellman equation.

2. Choose a policy satisfying (1.80).

Alternatively, one can specify a stationary policy, evaluate it and check whether it solves the Bellman equation.

Examples

Example 1.5. We revisit Example **??** and find a stationary deterministic optimal policy. We saw by direct calculation in Example 1.4 that when $\lambda = 0.9$ the stationary policy that used action $a_{1,2}$ in s_1 and $a_{2,2}$ in s_2 was discount optimal. We now investigate the impact of varying λ.

The optimality equations in component form are given by

$$v(s_1) = \max\{3 + \lambda(0.8v(s_1) + 0.2v(s_2)), 5 + \lambda v(s_2)\} \tag{1.82}$$
$$v(s_2) = \max\{-5 + \lambda v(s_2), 2 + \lambda(0.4v(s_1) + 0.6v(s_2))\}. \tag{1.83}$$

To solve this problem, we enumerate all possible decision rules, and then compute $v(s_1)$ and $v(s_2)$ for each one. Since there are two states and two actions in each state, there are four possible decision rules: $d_1 = (a_{1,1}, a_{2,1})$, $d_2 = (a_{1,1}, a_{2,2})$, $d_3 = (a_{1,2}, a_{2,1})$, $d_4 = (a_{1,2}, a_{2,2})$. Under d_1, we have

$$v_\lambda^{d_1^\infty}(s_1) = 3 + \lambda(0.8v_\lambda^{d_1^\infty}(s_1) + 0.2v_\lambda^{d_1^\infty}(s_2)) \tag{1.84}$$
$$v_\lambda^{d_1^\infty}(s_2) = -5 + \lambda v_\lambda^{d_1^\infty}(s_2). \tag{1.85}$$

Solving this system of equations we get

$$v_\lambda^{d_1^\infty}(s_1) = \frac{5(3-4\lambda)}{(1-\lambda)(5-4\lambda)} \tag{1.86}$$
$$v_\lambda^{d_1^\infty}(s_2) = \frac{-5}{(1-\lambda)}. \tag{1.87}$$

Repeating for the other policies, we get

$$v_\lambda^{d_2^\infty}(s_1) = \frac{15-7\lambda}{(1-\lambda)(5-2\lambda)} \tag{1.88}$$
$$v_\lambda^{d_2^\infty}(s_2) = \frac{2(5-\lambda)}{(1-\lambda)(5-2\lambda)}, \tag{1.89}$$

$$v_\lambda^{d_3^\infty}(s_1) = \frac{5(1-2\lambda)}{1-\lambda} \tag{1.90}$$
$$v_\lambda^{d_3^\infty}(s_2) = \frac{-5}{(1-\lambda)}, \tag{1.91}$$

$$v_\lambda^{d_4^\infty}(s_1) = \frac{5(5-\lambda)}{(1-\lambda)(5+2\lambda)} \tag{1.92}$$

$$v_\lambda^{d_4^\infty}(s_2) = \frac{10(1+\lambda)}{(1-\lambda)(5+2\lambda)}. \tag{1.93}$$

For $0 \leq \lambda < 1$, it is straightforward to show the following inequalities hold:

$$v_\lambda^{d_1^\infty}(s_1) < v_\lambda^{d_3^\infty}(s_1) \tag{1.94}$$

$$v_\lambda^{d_2^\infty}(s_1) < v_\lambda^{d_4^\infty}(s_1) \tag{1.95}$$

So, action $a_{1,2}$ is the optimal action in state s_1. Similarly, since $v_\lambda^{d_1^\infty}(s_2) = v_\lambda^{d_3^\infty}(s_2) < 0$, and both $v_\lambda^{d_2^\infty}(s_2) > 0$ and $v_\lambda^{d_4^\infty}(s_2) > 0$, for $0 \leq \lambda < 1$, action $a_{2,2}$ is optimal in state s_2. Thus, d_4^∞ is the optimal policy.

Alternatively we could check which of the four value functions satisfies the Bellman equation. We leave it as an exercise to show that $\mathbf{v}_\lambda^{d_4^\infty}$ does so.

Note in this example that the optimal policy does not depend on the specific value of λ. However, this is usually not the case. Hence, solving the problem algebraically may require defining intervals in which λ must lie for a certain policy to be optimal, as shown in the next example.

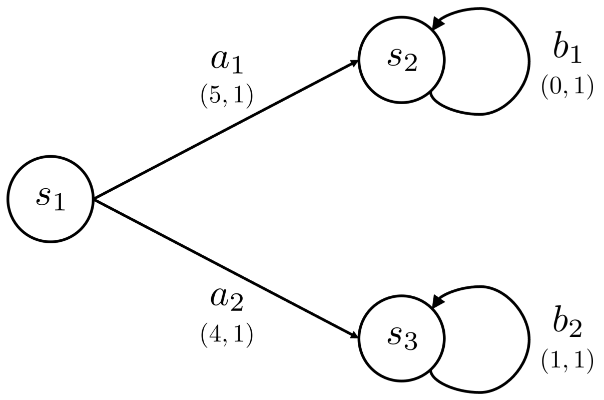


Figure 1.2: The three-state Markov decision process from Example 1.6 to illustrate the effect of varying λ on the optimal policy.

Example 1.6. 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 1.2. 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 optimality equations for this model are given by:

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

Therefore

$$v(s_1) = \max\left\{5, 4 + \frac{\lambda}{1-\lambda}\right\} = \begin{cases} 5 & 0 \leq \lambda \leq 0.5 \\ 4 + \lambda/(1-\lambda) & 0.5 \leq \lambda < 1 \end{cases}$$

Thus, the policy that uses action a_1 in state s_1 is optimal when if $0 \leq \lambda \leq 0.5$ and the policy that uses action a_2 in state s_1 is optimal when $0.5 \leq \lambda < 1$. Note that either action is 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.

In demonstrating that there exists an optimal stationary policy in Theorem 1.9, we required $\mathbf{v}^*_\lambda$. However, we have not yet discussed how to compute $\mathbf{v}^*_\lambda$, and therefore, how to compute an optimal stationary policy. While the enumeration approach demonstrated in Example 1.5 may be feasible for small "toy" problems, it is not scalable to larger problems. Thus, in the following sections, we present algorithms that compute $\mathbf{v}^*_\lambda$ and a corresponding optimal stationary policy from scratch.

1.3.6 State-action value functions

In Section **??**, we introduced the concept of state-action value functions. They will not be central to this chapter, but will be fundamental when we consider simulation-based methods, so discussion here will be brief.

Results above are expressed in terms of value functions, $v(s)$. We now provide equivalent representations in terms of state-action value functions $q(s,a)$. For the discounted model, we define the state-action value function corresponding to a specific policy π, $q^\pi(s,a)$, and the optimal state-action value function, $q^*(s,a)$, as follows:

$$q^\pi(s,a) := r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v^\pi_\lambda(j) \tag{1.96}$$

and

$$q^*(s,a) := r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_\lambda^*(j) \tag{1.97}$$

for all $a \in A_s$ and $s \in S$.

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

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

and from the Bellman equation (1.51), we have that

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

Hence, the Bellman equation expressed in terms of state-action value functions becomes

$$\boxed{q^*(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) \max_{a \in A_s} q^*(j,a). \tag{1.98}}$$

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

Written as an expected value, (1.98) is equivalent to

$$q^*(s,a) = E^{d^*}\left[r(X_1,a) + \lambda \max_{a \in A_s} q^*(X_2,a) \,\middle|\, X_1 = s \right], \tag{1.99}$$

where $(d^*)^\infty$ is an optimal policy for the discounted model. The previous equation extends to case where $r(s,a,j)$ is the model primitive as follows:

$$q^*(s,a) = E^{d^*}\left[r(X_1,a,X_2) + \lambda \max_{a \in A_s} q^*(X_2,a) \,\middle|\, X_1 = s \right]. \tag{1.100}$$

Observe that these two expressions differ from the usual Bellman equation in that the maximization is *inside* the expectation. While this difference is unimportant in this chapter, it will have a significant impact in Chapter **??**, where expectations are estimated by random draws from the transition probability distribution.

Contraction Mappings*

Let Q denote the set of real 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 $\mathbf{q} \in Q$, 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 $L : V \to V$ we define the (s,a)-th component of $F : Q \to Q$ by

$$F\mathbf{q}(s,a) := r(s,a) + \lambda \sum_{j \in S} p(j|s,a) \max_{a \in A_s} q(j,a). \tag{1.101}$$

We show that this operator is a contraction mapping on Q so that by the Banach fixed point theorem there exists a unique solution to $F\mathbf{q} = \mathbf{q}$ in Q.

Theorem 1.10. Suppose $0 \leq \lambda < 1$. Then the operator F defined in (1.101) is a contraction mapping on Q with modulus λ.

Proof. Let $\mathbf{q}_1$ and $\mathbf{q}_2$ be elements of Q. Then[8]

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

□

1.4 Spans, bounds and other technical concepts

To formally state an iterative algorithm for solving a discounted Markov decision process, we need:

- A stopping criterion
- An estimate of the optimal value function at termination
- An estimate of the optimal policy at termination

The previous sections have already addressed the third item. In this section, we focus on the first two, which will also be useful in the average reward and expected total reward models, and could be incorporated in simulation-based algorithms.

Suppose we have a sequence of iterates $\{\mathbf{v}^1, \mathbf{v}^2, \ldots\}$ generated by an iterative algorithm. We would like to know when these vectors are sufficiently close to each other so as to terminate the algorithm. We have already introduced the concept of a norm,

[8]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. We leave its proof as an exercise.

however the *span semi-norm* (defined below) provides enhanced convergence. Bounds tell us how well a stationary policy based on the "current" decision rule approximates the optimal policy and moreover provide a good estimate of the optimal value function. Bounds are also useful for identifying sub-optimal actions so that they need not be evaluated further.

1.4.1 Spans and span contractions

We introduce the span semi-norm[9] denoted by $\mathrm{sp}(\mathbf{v})$ and defined by

$$\mathrm{sp}(\mathbf{v}) = \max_{s\in S} v(s) - \min_{s\in S} v(s). \tag{1.102}$$

In addition to satisfying the properties of a semi-norm, for $\mathbf{v} \in V$:

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

We leave it as an exercise to prove that $\mathrm{sp}(\cdot)$ is a semi-norm and also satisfies the above properties. The first property means that when $\mathrm{sp}(\mathbf{v}) = 0$, $\mathbf{v}$ is a constant vector. On the other hand, when $\|\mathbf{v}\| = 0$, $\mathbf{v} = 0$. Hence the span semi-norm and the sup-norm measure different properties of a vector.

Span contractions

We state the most relevant property of the span in the following proposition.[10]

Proposition 1.2. For any $d \in D^{\mathrm{MR}}$, transition probability matrix $\mathbf{P}_d : S \to S$, and $\mathbf{v} \in V$,

$$\mathrm{sp}(\mathbf{P}_d\mathbf{v}) \leq \gamma_d \mathrm{sp}(\mathbf{v}), \tag{1.103}$$

[9]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 $\mathbf{v} = \mathbf{0}$.

[10]This result is Proposition 6.6.1 in Puterman [1994]. The proof can be found there.

where

$$\begin{aligned}\gamma_d = \ & 1 - \min_{(s,u)\in S\times S} \sum_{j\in S} \min\{P_d(j|s), P_d(j|u)\} \\ = \ & \frac{1}{2} \max_{(s,u)\in S\times S} \sum_{j\in S} |P_d(j|s) - P_d(j|u)| \\ = & \max_{(s,u)\in S\times S} \sum_{j\in S} (P_d(j|s) - P_d(j|u))^+. \end{aligned} \tag{1.104}$$

Moreover, there exists a $\mathbf{v} \in V$ for which (1.103) holds with equality.

The quantity γ_d is referred to as the *delta coefficient* or *Hajnal measure* of $\mathbf{P}_d$. It provides an upper bound on the second largest eigenvalue, referred to as the *sub-radius* of $\mathbf{P}_d$. Note that the above formulae are not intended for computation, but rather provide insight into the interpretation of γ_d. For example, when $\mathbf{P}_d = \mathbf{I}$ or for that matter when $\mathbf{P}_d$ has exactly one 1 in each row and column, it follows easily from each of these definitions that $\gamma_d = 1$. Note that γ'_d provides a simple upper bound on γ_d:

$$\gamma_d \le \gamma'_d := 1 - \sum_{j\in S} \min_{s\in S} P_d(j|s). \tag{1.105}$$

From Lemma 1.1, we have that $\|\mathbf{Pv}\| \le \|\mathbf{v}\|$ for any transition probability matrix on S. Thus, when $\gamma < 1$, $\mathbf{P}$ has contraction properties with respect to the span but not with respect to the norm.

Example 1.7. We now compute γ_d for the Markov deterministic decision rule $d(s_1) = a_{1,1}$, $d(s_2) = a_{2,1}$ in Example **??**.

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 $\mathrm{sp}(\mathbf{P}_d\mathbf{v}) = |0.8v_1 - 0.8v_2| = 0.8|v_1 - v_2| = 0.8\mathrm{sp}(\mathbf{v})$. Thus $\gamma_d = 0.8$. Note that for this transition probability matrix, the eigenvalues are 1 and 0.8 and that γ_d equals the second largest eigenvalue of $\mathbf{P}_d$.[a] The upper bound $\gamma' = 1 - (0 + 0.2) = 0.8$.

We leave it as an exercise 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 we find the second eigenvalue by subtracting 1 from its trace.

Note that when $\mathbf{P} = \mathbf{I}$, $\text{sp}(\mathbf{I}\mathbf{v}) = \text{sp}(\mathbf{v})$ for all $\mathbf{v} \in V$, so that $\gamma = 1$. The following proposition is one of the few results for a discounted model that depend on probabilistic properties of $\mathbf{P}_d$. We state it without proof. It is a consequence of the Perron-Frobenius Theorem, which appears as Theorem **??** in the Appendix.

Proposition 1.3. 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 **??** for definitions of the concepts below.

Observe that the transition probability matrix from Example 1.7 satisfies the second condition in Proposition 1.3.

As a consequence of Proposition 1.2 and the argument used to prove Proposition 1.1, the operator L is a contraction mapping with respect to the span semi-norm.

Theorem 1.11. For any $\mathbf{v}$ and $\mathbf{u}$ in V, and $0 \leq \lambda < 1$, there exists $\gamma^* \leq 1$ such that

$$\text{sp}(L\mathbf{v} - L\mathbf{u}) \leq \lambda\gamma^*\text{sp}(\mathbf{v} - \mathbf{u}). \tag{1.106}$$

By restricting the set of possible decision rules, we have the following corollary.

Corollary 1.3. For any $d \in D^{\text{MD}}$, $\mathbf{v}$ and $\mathbf{u}$ in V, and $0 \leq \lambda < 1$,

$$\text{sp}(L_d\mathbf{v} - L_d\mathbf{u}) \leq \lambda\gamma_d\text{sp}(\mathbf{v} - \mathbf{u}), \tag{1.107}$$

where γ_d is defined in (1.104).

Note that we cannot compute γ^* a priori. However, if either of the conditions in Proposition 1.3 hold, we know that $\gamma^* < 1$. Hence, in this case **the contraction modulus with respect to the span semi-norm is strictly less than that of the sup-norm** (i.e., $\lambda\gamma^* < \lambda$). This means that iterative algorithms such as value iteration (described in Section 1.5) will terminate faster when using the span as a stopping criterion.

We conclude this section by comparing the span and sup-norm contraction properties of L_d with an example.

Example 1.8. Let d be as in Example 1.7. Then

$$L_d\mathbf{v} - L_d\mathbf{u} = \mathbf{r}_d + \lambda\mathbf{P}_d\mathbf{v} - \mathbf{r}_d - \lambda\mathbf{P}_d\mathbf{u} = \lambda\mathbf{P}_d(\mathbf{v} - \mathbf{u}).$$

Choosing $\mathbf{v} = (2,4)$ and $\mathbf{u} = (5,1)$, $\|\mathbf{v} - \mathbf{u}\| = 3$ and $\text{sp}(\mathbf{v} - \mathbf{u}) = 6$. Since $\mathbf{P}_d\mathbf{v} = (2.4, 4)$ and $\mathbf{P}_d\mathbf{u} = (4.2, 1)$, $\mathbf{P}_d\mathbf{v} - \mathbf{P}_d\mathbf{u} = (-1.8, 3)$. Setting $\lambda = 0.5$ implies

$$L_d\mathbf{v} - L_d\mathbf{u} = \begin{bmatrix} -0.9 \\ 1.5 \end{bmatrix}$$

so that

$$\|L_d\mathbf{v} - L_d\mathbf{u}\| = 1.5 = 0.5\|\mathbf{v} - \mathbf{u}\|$$

and

$$\text{sp}(L_d\mathbf{v} - L_d\mathbf{u}) = 2.4 = (0.5)(0.8)\text{sp}(\mathbf{v} - \mathbf{u}).$$

Hence, the contraction modulus of the sup norm is 0.5 and that of the span is 0.4, as indicated by Corollary 1.3.

1.4.2 Bounds

We will primarily use bounds to determine how accurately $\mathbf{v}^n$ (the n-th iterate of an algorithm) approximates $\mathbf{v}^*_\lambda$ when an algorithm terminates. They will also provide the basis for action elimination procedures in Section 1.4.4.

We introduce some additional notation to simplify exposition. For $\mathbf{v} \in V$, define

$$\underline{\mathbf{v}} := \min_{s \in S} v(s) \quad \text{and} \quad \overline{\mathbf{v}} := \max_{s \in S} v(s). \tag{1.108}$$

Note that $\text{sp}(\mathbf{v}) = \overline{\mathbf{v}} - \underline{\mathbf{v}}$. We emphasize that $\overline{\mathbf{v}}$ and $\underline{\mathbf{v}}$ are scalars. Before stating our main result concerning bounds, we start with a definition and a technical lemma.

Definition 1.9. For any $\mathbf{v} \in V$, we say that $d_\mathbf{v} \in D^{\text{MD}}$ is $\mathbf{v}$-*improving* if

$$\mathbf{r}_{d_\mathbf{v}} + \lambda\mathbf{P}_{d_\mathbf{v}}\mathbf{v} = L\mathbf{v}. \tag{1.109}$$

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

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

Properties of $B\mathbf{v} := L\mathbf{v} - \mathbf{v}$

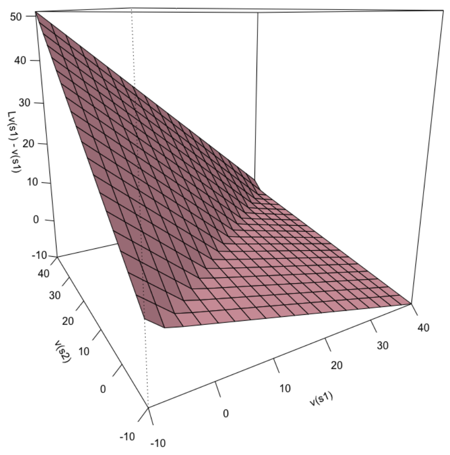
The quantity $L\mathbf{v} - \mathbf{v}$ plays an important role in deriving bounds, eliminating actions and proving convergence of several iterative algorithms. Since it arises so frequently we formally define the operator $B : V \to V$ by

$$B\mathbf{v} := L\mathbf{v} - \mathbf{v}. \tag{1.110}$$

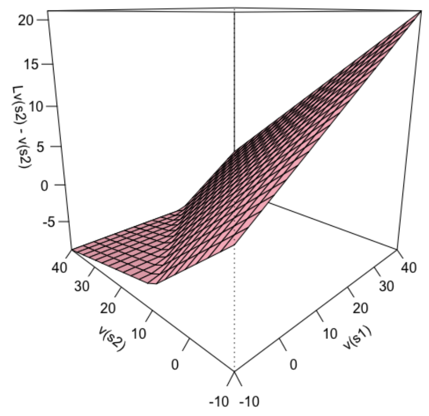
Note that the equation $L\mathbf{v} = \mathbf{v}$ is equivalent to $B\mathbf{v} = \mathbf{0}$ so that finding a fixed point of L is equivalent to finding a zero of B. For completeness, we define the operator B_d on V by $B_d := L_d\mathbf{v} - \mathbf{v}$. Thus, for consistency we can write

$$B\mathbf{v} = \operatorname*{c-max}_{d \in D^{\text{MD}}} B_d\mathbf{v}. \tag{1.111}$$

Figure 1.3 shows *each* component of $B\mathbf{v}$ as a function of the components of $\mathbf{v}$ for the two-state model in Section **??**. Because there are only two states, we can view the components of $\mathbf{v}$ on the horizontal axes and the value of *each* component of $B\mathbf{v}$ on the vertical axis. Observe that each component is piecewise linear and convex as a function of $\mathbf{v}$. Moreover $B\mathbf{v}(s_1)$ is non-increasing 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 $v(s_1)$.



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

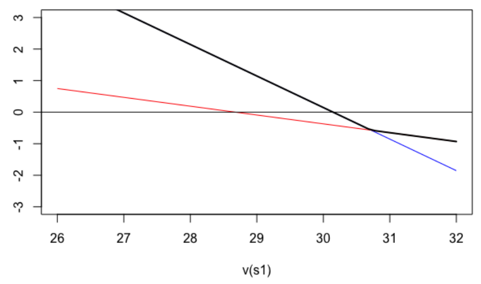


(b) $B\mathbf{v}(s_2)$ as a function of $v(s_1)$ and $v(s_2)$.

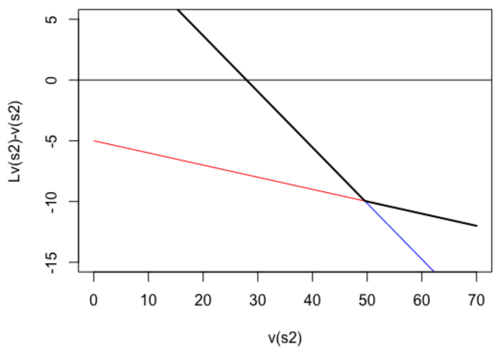
Figure 1.3: Components of $B\mathbf{v}$ as a function of the components of $\mathbf{v}$ for the two-state example.

To get a clearer view of what is being shown in Figure 1.3, we provide one-dimensional plots of $B\mathbf{v}(s_1)$ as a function of $v(s_1)$ for a fixed value of $v(s_2)$ and $B\mathbf{v}(s_2)$ as a function of $v(s_2)$ for a fixed value of $v(s_1)$ in Figure 1.4.

The following lemma, which generalizes the gradient inequality for convex func-



(a) The red line corresponds to action $a_{1,1}$, the blue line to action $a_{1,2}$ and the solid black line to $B\mathbf{v}(s_1)$ with $v(s_2) = 27.94$. Observe that $v(s_1) = 30.14$ when $B\mathbf{v}(s_1) = 0$ corresponding to action $a_{1,2}$.



(b) The red line corresponds to action $a_{2,1}$, the blue line to action $a_{2,2}$ and the solid black line to $B\mathbf{v}(s_2)$ with $v(s_1) = 30.14$. Observe that $v(s_2) = 27.94$ when $B\mathbf{v}(s_2) = 0$ corresponding to action $a_{2,2}$.

Figure 1.4: Plots of components of $B\mathbf{v}(s_1)$ as a function of $v(s_1)$ for $v(s_2)$ fixed at optimal value and $B\mathbf{v}(s_2)$ as a function of $v(s_2)$ for $v(s_1)$ fixed at optimal value.

tions[11], establishes the convexity properties of B, which we express both in terms of B and L.

Lemma 1.5. For any $\mathbf{v}$ and $\mathbf{u}$ in V and $\mathbf{v}$-improving $d_{\mathbf{v}}$,

$$B\mathbf{u} \geq B\mathbf{v} + (\lambda \mathbf{P}_{d_{\mathbf{v}}} - \mathbf{I})(\mathbf{u} - \mathbf{v}), \tag{1.112}$$

or equivalently,

$$L\mathbf{u} - \mathbf{u} \geq L\mathbf{v} - \mathbf{v} + (\lambda \mathbf{P}_{d_{\mathbf{v}}} - \mathbf{I})(\mathbf{u} - \mathbf{v}). \tag{1.113}$$

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. □

[11]If $f(x)$ is a convex and differentiable function mapping $\Re^1$ into $\Re^1$, then for any x and y in $\Re^1$,

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

Theorem 1.12. For any $\mathbf{v} \in V$, and $\mathbf{v}$-improving $d_{\mathbf{v}} \in D^{\text{MR}}$

$$\begin{aligned} &\mathbf{v} + (1-\lambda)^{-1}(\overline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \geq L\mathbf{v} + \lambda(1-\lambda)^{-1}(\overline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v}_\lambda^* \geq \mathbf{v}_\lambda^{d_v^\infty} \\ &\qquad \geq L\mathbf{v} + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v} + (1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} \end{aligned} \tag{1.114}$$

Proof. We derive the lower bounds in (1.114). The upper bounds are obtained by reversing $\mathbf{v}$ and $\mathbf{v}_\lambda^*$ in the following.

Applying Lemma 1.5 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 1.3, 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

$$\begin{aligned} \mathbf{v}_\lambda^* &\geq \mathbf{v} + (\mathbf{I} - \lambda\mathbf{P}_{d_v})^{-1}(L\mathbf{v} - \mathbf{v}) &(1.115)\\ &= \mathbf{v} + (\mathbf{I} + \lambda\mathbf{P}_{d_{\mathbf{v}}} + \lambda^2\mathbf{P}_{d_{\mathbf{v}}}^2 + \ldots)(L\mathbf{v} - \mathbf{v}) &(1.116)\\ &= \mathbf{v} + (L\mathbf{v} - \mathbf{v}) + \lambda\mathbf{P}_{d_{\mathbf{v}}}(\mathbf{I} + \lambda\mathbf{P}_{d_{\mathbf{v}}} + \lambda^2\mathbf{P}_{d_{\mathbf{v}}}^2 + \ldots)(L\mathbf{v} - \mathbf{v}) &(1.117)\\ &\geq \mathbf{v} + (L\mathbf{v} - \mathbf{v}) + \lambda\mathbf{P}_{d_{\mathbf{v}}}(\mathbf{I} + \lambda\mathbf{P}_{d_{\mathbf{v}}} + \lambda^2\mathbf{P}_{d_{\mathbf{v}}}^2 + \ldots)(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} &(1.118)\\ &= \mathbf{v} + (L\mathbf{v} - \mathbf{v}) + \lambda\mathbf{P}_{d_{\mathbf{v}}}(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} &(1.119)\\ &= \mathbf{v} + (L\mathbf{v} - \mathbf{v}) + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e} &(1.120)\\ &\geq \mathbf{v} + (1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e}. &(1.121) \end{aligned}$$

Equality (1.116) is an immediate consequence of Theorem 1.3 and (1.117) follows by rewriting (1.116). Inequality (1.118) follows by noting $L\mathbf{v} - \mathbf{v}$ is lower bounded by $(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e}$. Equation (1.119) follows from the third part of Lemma 1.3. Equation (1.120) 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 (1.120) yields the second lower bound in (1.114). The loosest lower bound in (1.114) follows by replacing $L\mathbf{v} - \mathbf{v}$ in (1.120) by $(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e}$.

Finally, rewriting the right hand side of (1.115),

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

we have the tightest lower bound in (1.114). □

Some comments about the above bounds follow:

1. Several generalizations of these bounds are possible by splitting (1.117) at higher order terms, but those above are most appropriate for our immediate purposes.

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 we also need to evaluate $L\mathbf{v}$ to evaluate the looser outer bound.

We illustrate computation of these bounds in the context of our two-state example.

Example 1.9. From Example 1.5,

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

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

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

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 $\mathbf{v}$-improving. Thus, applying L to $\mathbf{v}$ increased the first component by 0.7 and the second component by 6.1.

Therefore, the bounds in (1.114) 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 $\mathbf{v}_\lambda^*$. We leave it as an exercise to evaluate these bounds for other choices of $\mathbf{v}$.

1.4.3 Stopping Criteria

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

Theorem 1.13. For any $\mathbf{v} \in V$ and $\epsilon > 0$, if

$$\mathrm{sp}(L\mathbf{v} - \mathbf{v}) < \frac{1-\lambda}{\lambda}\epsilon \tag{1.122}$$

then

$$\|L\mathbf{v} + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v}-\mathbf{v}})\mathbf{e} - \mathbf{v}_\lambda^*\| < \epsilon \tag{1.123}$$

and for any $\mathbf{v}$-improving $d_\mathbf{v} \in D^{\text{MR}}$

$$\|\mathbf{v}_\lambda^{d_\mathbf{v}^\infty} - \mathbf{v}_\lambda^*\| < \epsilon. \tag{1.124}$$

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 (1.123) and (1.124) follow from (1.114) 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 $\text{sp}(L\mathbf{v}-\mathbf{v}) = \overline{L\mathbf{v}-\mathbf{v}} - \underline{L\mathbf{v}-\mathbf{v}})$ and $\|\mathbf{e}\| = 1$. □

Since in most iterative algorithms we will not have 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 $\mathbf{v}_\lambda^*$. However even without using this estimate, we know that $d_\mathbf{v}^\infty$ is ϵ-optimal although we may not know its value without further calculations.

1.4.4 Action elimination

We begin with the following obvious result which provides the basis for eliminating non-optimal actions.

Proposition 1.4. 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

$$r(s,a') + \lambda \sum_{j \in S} p(j|s,a') v_\lambda^*(j) < \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_\lambda^*(j) \right\}. \tag{1.125}$$

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

See Figure 1.4 for a graphical illustration of this result in the context of the two-state model of Section **??**. In the left hand figure, the red line corresponding to $a' = a_{1,1}$ satisfies (1.125) in s_1 and in the right hand figure, the red line corresponding to action $a' = a_{2,1}$ satisfies (1.125) in s_2.

Since we do not know $\mathbf{v}_\lambda^*$, we cannot implement (1.125) directly to identify non-optimal actions. However we can use bounds on $\mathbf{v}_\lambda^*$ to provide an approach that can be put in practice as follows.

Proposition 1.5. Suppose $\mathbf{v}^L \leq \mathbf{v}_\lambda^* \leq \mathbf{v}^U$. Then if

$$r(s,a') + \lambda \sum_{j\in S} p(j|s,a') v^U(j) < v^L(s) \tag{1.126}$$

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

Proof. Suppose (1.126) 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 1.4 □

We can now use the bounds in the previous section to provide an explicit rule to eliminate non-optimal actions. We substitute the outer upper bound and inner lower bound from (1.114) into (1.126) to obtain for any $\mathbf{v} \in V$,

$$r(s,a') + \lambda \sum_{j\in S} p(j|s,a')\left(v(j) + (1-\lambda)^{-1}(\overline{L\mathbf{v}-\mathbf{v}})\right) < L\mathbf{v}(s) + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v}-\mathbf{v}}). \tag{1.127}$$

Reorganizing terms leads to the main result of this section.

Corollary 1.4. For any $\mathbf{v} \in V$ if

$$\frac{\lambda}{(1-\lambda)}\mathrm{sp}(L\mathbf{v}-\mathbf{v}) < L\mathbf{v}(s) - \left(r(s,a') + \lambda \sum_{j\in S} p(j|s,a') v(j)\right), \tag{1.128}$$

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

Some comments follow:

1. Note that our choice of bounds in (1.127) 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 we had instead used the tighter upper bound.
2. By organizing calculations efficiently, the right hand side of (1.128) can be evaluated with minimal extra effort.
3. We will show how to integrate this result into value iteration in the next section.
4. The literature also includes action elimination methods that identify actions for temporary elimination, that is, they will not be part of a $\mathbf{v}^n$-improving decision rule, where $\mathbf{v}^n$ is the n-th iterate of the value function in an algorithm, but could be in subsequent iterations.

We conclude this section with an example.

Example 1.10. We continue with the calculations in Example 1.9. 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 (1.128). We 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 (1.128). 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 (1.128) 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 $\mathbf{v} = \mathbf{v}^5 = (13.24, 11.06)$ that action $a_{1,1}$ is eliminated in s_1. Hence after six iterations of $\mathbf{v}^{n+1} = L\mathbf{v}^n$, we have found 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 $\mathbf{v}^*_\lambda = (30.147, 27.941)$. We conclude by noting that value obtained by the lower bound extrapolation in (1.123) equals $(30.08, 27.87)$, which is a close approximation to $\mathbf{v}^*_\lambda$.

This example suggests how to incorporate action elimination in an iterative procedure. It shows that when all but one action in each state has been eliminated, then we have found the optimal policy. While they may work well in certain situations, in our experience, action elimination methods are not always so rewarding.

1.5 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 $f(x) = x$.

1.5.1 The value iteration algorithm

Except for some details regarding estimation of the optimal value function at termination the following gives a high level view of the essence of the value iteration algorithm.

Value iteration: vector notation

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

In practice value iteration is applied component-wise as follows. Iterates of the algorithm will be denoted using a superscript on the value function to distinguish them from our use of subscripts to denote decision epochs.

Value iteration: component notation

1. **Initialize:** Set $n = 0$, specify $\epsilon > 0$ and $v^0(s)$ for all $s \in S$.

2. **Iterate:** Compute $v^{n+1}(s)$ for all $s \in S$:

$$v^{n+1}(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v^n(j) \right\}. \tag{1.129}$$

3. **Apply stopping criterion:** If

$$\mathrm{sp}(\mathbf{v}^{n+1} - \mathbf{v}^n) \geq \frac{\lambda}{1-\lambda}\epsilon,$$

proceed to step 4, else $n \leftarrow n+1$ and return to step 2.

4. **Choose an ϵ-optimal policy:** For all $s \in S$, choose

$$d^\epsilon(s) \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v^{n+1}(j) \right\}$$

and

$$v^\epsilon(s) = v^n(s) + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}}).$$

Before proving convergence, we offer several comments:

1. The iterative step (1.129) represents the essence of the algorithm.

2. Judicious choice of a starting value can enhance convergence. One good choice is $v^0(s) = \max_{a \in A_s} r(s,a)$.

3. 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 (1.129) will only contain a few terms. For example, in our queuing rate control model (Section **??**), there will be at most three non-zero probabilities, enabling the sum to be easily coded and quickly computed.

4. We use the span (as opposed to the norm) as the basis for terminating the algorithm. Section 1.4.1 suggests that the span-based stopping criterion will be satisfied in fewer iterations than the norm-based stopping criterion. We will explore this comparison through numerical examples below.

5. When the stopping criterion is satisfied, specifying $d^\epsilon(s)$ requires no addition computation. It can be determined while computing (1.129). Theorem 1.13 ensures that it is ϵ-optimal. Note that its value is not determined by the algorithm.

6. The expression for $v^\epsilon(s)$, referred to as the lower bound extrapolation in Section 1.4.3, can be determined with little extra computational effort. We use it instead of $\mathbf{v}^{n+1}$ to approximate $\mathbf{v}^*_\lambda$ because Theorem 1.13 ensures that each component of the lower bound extrapolation will be within ϵ of optimal.

7. Action elimination can be used to enhance the algorithm. We will show how to do so below.

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

9. 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 1.14. Suppose $0 \leq \lambda < 1$ and $\epsilon > 0$. Let the sequence $\mathbf{v}^n, n = 1, 2, \ldots$ be generated by value iteration, i.e., $\mathbf{v}^n = L^n \mathbf{v}^0$ for any $\mathbf{v}^0 \in V$. Then:

1. $\|\mathbf{v}^n - \mathbf{v}^*_\lambda\| \to 0$ as $n \to \infty$,

2. d_ϵ^∞ is ϵ-optimal, and

3. $\|\mathbf{v}^\epsilon - \mathbf{v}^*_\lambda\| < \epsilon$.

Proof. Proposition 1.1 establishes that L is a contraction mapping. Hence by Theorem 1.5, $\mathbf{v}^n$ converges to a fixed point of L. From Theorem 1.8, the limit is unique and equals $\mathbf{v}^*_\lambda$. Parts 2 and 3 are restatements of Theorem 1.13. □

We now solve the two-state model from Section **??** using value iteration. We compare the rate of decrease of the span and the norm.

Example 1.11. We set $\lambda = 0.9$ and $\epsilon = 0.000001$. We choose $\mathbf{v}^0 = \mathbf{0}$. Using $\|\mathbf{v}^{n+1} - \mathbf{v}^n\|$, the algorithm terminates after 162 iterations and using $\mathrm{sp}(\mathbf{v}^{n+1} - \mathbf{v}^n)$, 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 1.5 compares the convergence of $\mathbf{v}^{n+1} - \mathbf{v}^n$ with respect to these two measures.

At termination using the span criterion, $\mathbf{v} = (25.39, 23.19)$ and $\mathbf{v}^\epsilon = (30.15, 27.94)$. Using the norm criterion, we find that $\mathbf{v} = \mathbf{v}^\epsilon = (30.15, 27.94)$. Thus in this example we conclude that the span converges 10 times faster than the norm and the lower bound extrapolation produces identical estimates of the optimal value function.

We leave it as exercise to investigate the impact of $\mathbf{v}^0$ on convergence.

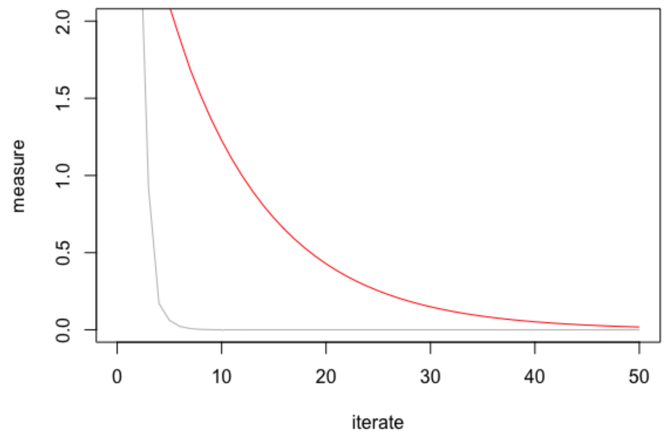


Figure 1.5: Comparison of convergence rate of $\mathbf{v}^{n+1} - \mathbf{v}^n$ with respect to the span (grey) and norm (red).

1.5.2 Gauss-Seidel iteration

Since we implement value iteration component-wise, assuming that the states are updated in consecutive order $s_1, s_2, \ldots, s_M$, then the values $v^{n+1}(s_j)$ for $j < k$ have already been calculated before we evaluate $v^{n+1}(s_k)$. Gauss-Seidel iteration accounts for this observation by using the following value function update in state s_k

$$v^{n+1}(s_k) = \max_{a \in A_{s_k}} \left\{ r(s_k, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) v^{n+1}(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v^n(s_j) \right) \right\}. \tag{1.130}$$

Thus, it uses the most current estimate of the value function in all states when doing its update. One would expect faster convergence by doing so.

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

$$G\mathbf{v}(s_k) := \max_{a \in A_{s_k}} \left\{ r(s_k, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) G\mathbf{v}(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v(s_j) \right) \right\}. \tag{1.131}$$

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. Recall that throughout this section we assume states are labelled $s_1, \dots, s_M$ and are evaluated in that order in the following algorithm.

Gauss-Seidel iteration

1. **Initialize:** Select $\epsilon > 0$, $\mathbf{v}^0 \in V$ and set $n = 0$.
2. **Iterate:** For $k = 1, \dots, M$ compute
$$v^{n+1}(s_k) = \max_{a \in A_s} \left\{ r(s, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) v^{n+1}(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v^n(s_j) \right) \right\}. \tag{1.132}$$
3. **Check stopping criterion:** If $\|\mathbf{v}^{n+1} - \mathbf{v}^n\| < (1-\lambda)^{-1}\epsilon$ proceed to step 4, else $n \leftarrow n + 1$ and return to step 2.
4. **Terminate:** For $k = 1, \dots, M$, select $d^\epsilon(s_k)$ by
$$d^\epsilon(s_k) \in \arg\max_{a \in A_s} \left\{ r(s, a) + \lambda \sum_{j \in S} p(s_j|s_k, a) v^{n+1}(s_j) \right\}. \tag{1.133}$$

We provide some comments on the algorithm.

1. This algorithm uses a norm-based stopping criterion since the bounds used to derive the span-based criterion for value iteration are not available.
2. The ordering of the states is immaterial and can vary from iteration to iteration. The analysis in this chapter is valid as long as there is a complete pass through all states at each iteration.

Convergence of Gauss-Seidel iteration

We prove that Gauss-Seidel iteration converges to $\mathbf{v}_\lambda^*$ by showing that the updates form a contraction mapping and then applying the Banach fixed point theorem. We now modify the proof of Proposition 1.1 to show that G is a contraction.

Proposition 1.6. Suppose $G : V \to V$ is defined by (1.131) and $0 \leq \lambda < 1$. Then G is a contraction mapping with modulus λ with respect to the sup-norm.

Proof. Choose $\mathbf{v}$ and $\mathbf{u}$ in V. That

$$|G\mathbf{v}(s_1) - G\mathbf{u}(s_1)| \leq \lambda \|\mathbf{v} - \mathbf{u}\| \tag{1.134}$$

uses the same argument in the proof of Proposition 1.1, since $G\mathbf{v}(s_1) = L\mathbf{v}(s_1)$. We complete the proof by induction on the index of the state.

Assume (1.134) holds when s_j replaces s_1 for $j = 1, \dots, k-1$. Suppose $G\mathbf{v}(s_k) \geq G\mathbf{u}(s_k)$ and let $a' \in A_{s_k}$ such that

$$r(s_k, a') + \lambda \left(\sum_{j<k} p(s_j|s_k, a') G\mathbf{v}(s_j) + \sum_{j \geq k} p(s_j|s_k, a') v(s_j) \right) \tag{1.135}$$

$$= \max_{a \in A_s} \left\{ r(s_k, a) + \lambda \left(\sum_{j<k} p(s_j|s_k, a) G\mathbf{v}(s_j) + \sum_{j \geq k} p(s_j|s_k, a) v(s_j) \right) \right\} \tag{1.136}$$

$$= G\mathbf{v}(s_k). \tag{1.137}$$

Then by a similar argument to that used to prove Proposition 1.1

$$\begin{aligned}
0 &\leq G\mathbf{v}(s_k) - G\mathbf{u}(s_k) \\
&= r(s_k, a') + \lambda \left(\sum_{j<k} p(j|s_k, a') G\mathbf{v}(s_j) + \sum_{j \geq k} p(j|s_k, a') v(s_j) \right) \\
&\quad - \left\{ r(s_k, a') + \lambda \left(\sum_{j<k} p(s_j|s_k, a') G\mathbf{u}(s_j) + \sum_{j \geq k} p(s_j|s_k, a') u(s_j) \right) \right\} \\
&\leq \lambda \left(\sum_{j<k} p(s_j|s_k, a') |G\mathbf{v}(s_j) - G\mathbf{u}(s_j)| + \sum_{s_j \geq k} p(j|s_k, a') \|\mathbf{v} - \mathbf{u}\| \right) \leq \|\mathbf{v} - \mathbf{u}\|,
\end{aligned} \tag{1.138}$$

where (1.138) follows from the induction hypothesis and noting since $\lambda < 1$, $|G\mathbf{v}(s_j) - G\mathbf{u}(s_j)| \leq \lambda \|\mathbf{v} - \mathbf{u}\| \leq \|\mathbf{v} - \mathbf{u}\|$.

Repeating the above argument for when $G\mathbf{u}(s_k) \geq G\mathbf{v}(s_k)$ establishes that (1.134) 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 of value iteration since we do not have bounds like (1.114) in terms of $G\mathbf{v}$.

Theorem 1.15. Suppose $0 \leq \lambda < 1$ and $\epsilon > 0$. Let the sequence $(\mathbf{v}^n), n = 1, 2, \dots$ be generated by Gauss-Siedel iteration, i.e., $\mathbf{v}^n = G^n \mathbf{v}^0$ for any $\mathbf{v}^0 \in V$. Then:

1. $\|\mathbf{v}^n - \mathbf{v}^*_\lambda\| \to 0$ as $n \to \infty$,

2. d_ϵ^∞ is ϵ-optimal,

3. $\|\mathbf{v}^n - \mathbf{v}^*_\lambda\| < \epsilon$ and $\mathrm{sp}(\mathbf{v}^n - \mathbf{v}^*_\lambda) < 2\epsilon$.

Proof. Since G is a contraction, $G^n\mathbf{v}_0$ converges to a fixed point $\mathbf{v}^*$ by Theorem 1.5. Since $G\mathbf{v}^* = \mathbf{v}^*$, from (1.131),

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

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

Hence, $\mathbf{v}^*$ is a fixed point of L. Since L has a unique fixed point $\mathbf{v}^*_\lambda$ the first result follows.

By construction, d_ϵ is $\mathbf{v}^{n+1}$-improving, so by Theorem 1.13, d_ϵ^∞ is ϵ-optimal.

Finally, 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 $(1-\lambda)\|\mathbf{v}^n - \mathbf{v}^*_\lambda\| \leq \|\mathbf{v}^{n+1} - \mathbf{v}^n\|$ from which the norm relationship in part 3 follows. The result for the span follows by the property that $\mathrm{sp}(\mathbf{v}) \leq 2\|\mathbf{v}\|$.

□

Example 1.12. We continue the two-state example from the preceding section. Using the same initialization as in Example 1.11 we apply Gauss-Seidel (GS) iteration. Using a sup-norm stopping criterion, the algorithm terminates in 120 iterations. Using the span-based criterion requires 98 iterations to satisfy the stopping criterion. Under both stopping criteria, the algorithm terminates with the estimate $\mathbf{v}^{n+1} = (30.15, 27.94) = \mathbf{v}^*_\lambda$ so that extrapolation is not beneficial. Thus in this example, the span offers little benefit over the sup-norm. This is because the bounds in Theorem 1.12 do not apply for Gauss-Seidel.

Figure 1.6 compares the convergence of value iteration and Gauss-Seidel value iteration showing the sup-norm for GS and value iteration and the span for GS. We note that $\|\mathbf{v}^{n+1} - \mathbf{v}^n\|/\|\mathbf{v}^n - \mathbf{v}^{n-1}\| \approx 0.86$ for GS, while that ratio equals 0.9 for value iteration.

A hybrid approach plus some discussion

Appendix 1.12.2 provides bounds and develops some interesting matrix representations for Gauss-Seidel iteration. As shown in Section 1.4.3, the span stopping criterion for value iteration is based on subtracting the lower bound from the upper bound as in Theorem 1.13. Since the lower bound for Gauss-Seidel cannot be evaluated easily,

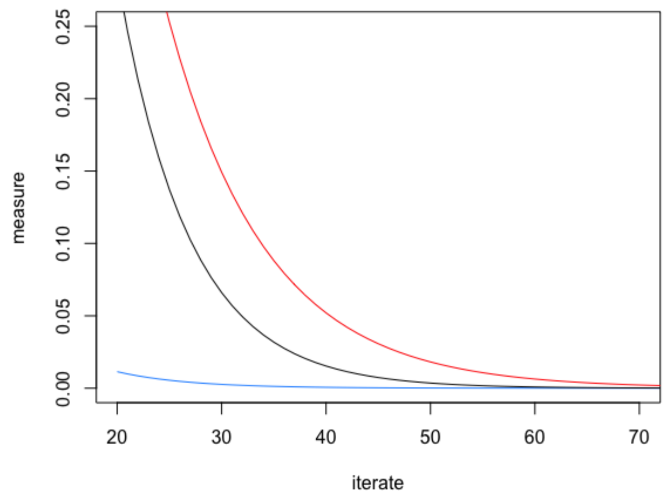


Figure 1.6: Comparison of convergence of Gauss-Seidel and value iteration for 2-state example. The black line shows $\|\mathbf{v}^{n+1} - \mathbf{v}^n\|$ for Gauss-Seidel, the red line shows $\|\mathbf{v}^{n+1} - \mathbf{v}^n\|$ for value iteration and the blue line shows $\text{sp}(\mathbf{v}^{n+1} - \mathbf{v}^n)$ for Gauss-Seidel. Note the axis scales when interpreting this figure.

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 either:

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

The advantage of the later approach is the availability of bounds, the applicability of the span stopping criterion and terminating 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

$$v^{n+1}(s) = \begin{cases} \max_{a \in A_s}\{r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v^n(j)\} & s = s' \\ v^n(s) & s \neq s'. \end{cases} \tag{1.139}$$

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 **??** and also updates generated by parallel computation. We refer you to the literature for more details and a proof of convergence.

1.5.3 Value iteration with action elimination

In this brief section, we show how to use the result in Corollary 1.4 to eliminate actions while using value iteration. It requires minimal extra computation but a minimal amount of storage.

Value iteration with action elimination

1. **Initialization:** Select $\epsilon > 0$, choose $\mathbf{v}^0 \in V$, set $A_s^0 = A_s$ for all $s \in S$ and $n = 0$.

2. **Update values:** For each $s \in S$, for all $a' \in A_s^n$ evaluate

$$q(s, a') = r(s, a') + \lambda \sum_{s \in S} p(j|s, a') v^n(j) \tag{1.140}$$

and set $L\mathbf{v}^n(s) = \max_{a \in A_s^n} q(s, a)$.

3. **Identify and eliminate non-optimal actions:**

Compute $\text{sp}(L\mathbf{v}^n - \mathbf{v}^n)$ and for each $s \in S$, set

$$A_s^{n+1} = \left\{ a' \in A_s^n \;\middle|\; \frac{\lambda}{(1-\lambda)} \text{sp}(L\mathbf{v}^n - \mathbf{v}^n) \geq L\mathbf{v}^n(s) - q(s, a') \right\} \tag{1.141}$$

4. **Apply stopping criterion:**

If

$$\text{sp}(L\mathbf{v}^n - \mathbf{v}^n) < \frac{\lambda}{(1-\lambda)} \epsilon$$

or A_s^{n+1} is a singleton for all $s \in S$, stop. Else set $\mathbf{v}^{n+1} = L\mathbf{v}^n$, $n \leftarrow n + 1$ and go to step 2.

5. **Extrapolation:**

For each $s \in S$ choose

$$d^\epsilon(s) \in \arg\max_{a \in A_s^n} q(s, a) \tag{1.142}$$

and

$$v^\epsilon(s) = v^n(s) + \frac{\lambda}{(1-\lambda)} (\underline{L\mathbf{v}^n - \mathbf{v}^n}). \tag{1.143}$$

Some comments on this algorithm follow:

1. The algorithm terminates with *either* an ϵ-optimal policy when terminated with the span criterion, or an optimal policy if A_s^{n+1} contains a single element for all $s \in S$. In the latter case, the optimal policy is unique. Note that this is the only way to implement value iteration so that it finds an optimal, as opposed to an ϵ-optimal policy. In both cases $\mathbf{v}^\epsilon$ is within ϵ of the optimal value function.

2. Note that it is necessary to evaluate $L\mathbf{v}^n(s)$ for all $s \in S$ before eliminating

actions. Therefore we separate steps 2 and 3.

3. 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 intermediate calculation in determining $L\mathbf{v}^n$. When not using action elimination, it need not be stored for any state.
4. Note that $A_s^{n+1} \subseteq A_s^n$ for all $s \in S$.
5. We do not apply the algorithm to Gauss-Seidel iteration because of the unavailability of easily computable lower bounds.

1.6 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. Moreover it can be easily adapted to simulation based methods, has several modifications that require less computational effort and may simplify implementation. 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. We will use the acronym PIA for policy iteration algorithm. While value iteration may only find an ϵ-optimal policy, policy iteration will find an optimal policy.

1.6.1 The algorithm

We first express policy iteration in vector-matrix notation to provide a high-level perspective on the key steps of the algorithm.

The Policy Iteration Algorithm (PIA): vector-matrix notation

1. **Initialization:** Choose $d' \in D^{\text{MD}}$.
2. **Evaluation:** Obtain $\mathbf{v}'$ by solving
$$(\mathbf{I} - \lambda \mathbf{P}_{d'})\mathbf{v} = \mathbf{r}_{d'}. \tag{1.144}$$
3. **Improvement:** Select
$$d'' \in \underset{d \in D^{\text{MD}}}{\arg\,\text{c-max}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}'\} \tag{1.145}$$
and set $d'' = d'$ if possible.
4. **Termination:** If $d'' = d'$, stop. Otherwise $d' \leftarrow d''$ and return to step 2.

Before stating it in a form suitable for computation, we provide some comments on the algorithm.

1. In contrast to value iteration, policy iteration begins with a stationary decision rule instead of a value. Alternatively, one can start in step 3 with any $\mathbf{v}' \in V$.

2. The evaluation step involves solving a system of linear equations, which requires at most $O(M^3)$ operations when $|S| = M$.

3. The improvement step requires the same effort as one pass through value iteration. We emphasize that it is implemented component-wise to avoid enumerating all Markovian deterministic decision rules. Modified policy iteration, described below, replaces this step with an iterative method for approximating the value function of $(d')^\infty$.

4. The instruction "set $d'' = d'$ if possible" prevents cycling when the maximizer is non-unique in some states. It is also fundamental to enable the stopping condition to be satisfied. Without this rule we can replace the stopping criterion by one based on the value functions being identical at two iterations. However, one must account for the numerical precision to which these quantities have been computed.

5. A stopping condition with bounds could replace the specified stopping condition based on obtaining the same maximizer at successive iterations.

6. An action elimination procedure can be incorporated into the PIA.

7. The next section shows that PIA 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.

We now restate PIA in component notation to provide a precise implementation process. This description emphasizes that the improvement step is implemented state by state.

The Policy Iteration Algorithm (PIA): component notation

1. **Initialization:** For each $s \in S$, choose a'_s for some $a'_s \in A_s$.
2. **Evaluation:** Obtain $v'(s)$ for all $s \in S$ by solving the system of linear equations
$$v(s) - \lambda \sum_{j \in S} p(j|s, a'_s) v(j) = r(s, a'_s). \tag{1.146}$$
3. **Improvement:** For each $s \in S$ select
$$a''_s \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v'(j) \right\} \tag{1.147}$$
and set $a''_s = a'_s$ if possible.
4. **Termination:** If $a''_s = a'_s$ for all $s \in S$, stop. Otherwise for all $s \in S$, $a'_s \leftarrow a''_s$ and return to step 2.

As noted above, if evaluation takes place as soon as there is a strict improvement in some state in step 3, that is for some s and $\delta > 0$

$$\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v'(j) \right\} = r(s, a'_s) + \lambda \sum_{j \in S} p(j|s, a'_s) v'(j) + \delta, \tag{1.148}$$

we can still show that the algorithm converges in the finite state and action model.

Example 1.13. We now solve the two-state example from Section **??** using policy iteration. We use the calculations in Example 1.4 for evaluation. We set $\lambda = 0.9$. Policy Iteration in component notation proceeds as follows:

1. Choose $a'_{s_1} = a_{1,2}$ and $a'_{s_2} = a_{2,1}$.
2. Evaluate it to find $v'(s_1) = -40$ and $v'(s_2) = -50$
3. In the improvement step we have
$$\begin{aligned} a''_{s_1} &= \arg\max_{i=1,2} \left\{ r(s_1, a_{1,i}) + \lambda \sum_{j=1,2} p(s_j|s_1, a_{1,i}) v'(s_j) \right\} \\ &= \arg\max\{3 + 0.9(0.8v'(s_1) + 0.2v'(s_2)), 5 + 0.9v'(s_2)\} \\ &= \arg\max\{-34.8, -40\} = a_{1,1} \end{aligned}$$

and

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

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 1.43, we obtain $v'(s_1) = 27.188$ and $v'(s_2) = 25.625$.

6. Applying the improvement step we find that

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

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. Appealing again to Example 1.43, we obtain $v'(s_1) = 30.147$ and $v'(s_2) = 27.941$.

9. Applying the improvement step we find that

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

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. 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.

Moreover note that the sequence of value functions is non-decreasing.

The above example is quite typical of policy iteration performance. It usually

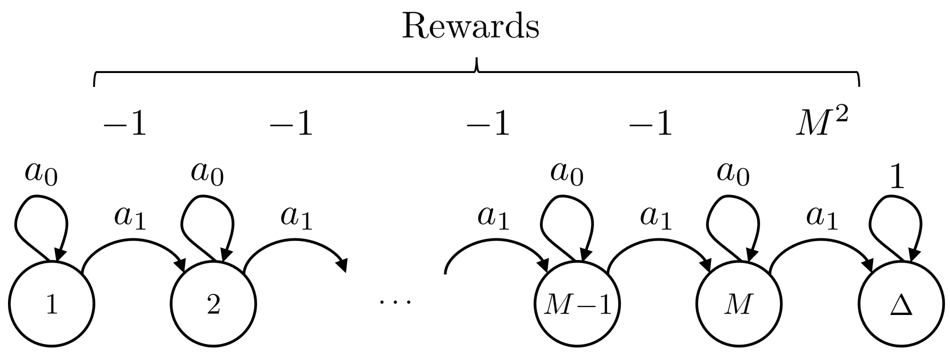


Figure 1.7: Figure for Example 1.14. Actions a_0 and a_1 indicate transitions with probability 1 given the chosen action. Rewards are associated with transitions to the "right".

requires a few iterations to find an optimal policy. However the following example shows that policy iteration may require $M + 1$ iterations to find an optimal policy where $|S| = M$.

Example 1.14. The following simple deterministic example shows that policy iteration may require M iterations to find an optimal policy. In all states except the last, the reward is zero. It costs 1 to go to the right except in state M where the reward is M^2 and there is a transition to a zero reward absorbing state Δ. See Figure 1.7.

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 $r(s, a_0) = 0,\ s \in S$, $r(s, a_1) = -1,\ s < M,\ r(M, a_1) = M^2$. The transition probabilities are $p(s|s, a_0) = 1$ for $s \in S$ and $p(s + 1|s, a_1) = 1$ for $s < M$.

We now implement policy iteration. Choose $a'_s = a_0$ for all $s \in S$, then $v'(s) = 0$ 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 all $s \in S$ and return to step 2 in PIA.

We find that $v'(s) = 0$ for $s < M$ and $v'(M) = M^2$. Then in the improvement step we obtain $a''_s = a_1$ for $s \in \{M - 1, M\}$ and $a''_s = a_0$ otherwise. We evaluate this policy to obtain

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

Continuing in this way we see that it requires $M + 1$ iterations to identify an optimal policy.

1.6.2 Convergence of Policy Iteration

In this section, we prove that policy iteration finds an optimal policy in a finite number of iterations. We first show that as we observed in the examples in the preceding section, the sequence of values generated by policy iteration is non-decreasing.

We begin with the result that a *strict* improvement leads to a strict increase in value. We use vector-matrix notation to simplify the proof.

Proposition 1.7. 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 $\delta > 0$

$$\mathbf{r}_{d''} + \lambda \mathbf{P}_{d''}\mathbf{v}' = \mathbf{r}_{d'} + \lambda \mathbf{P}_{d'}\mathbf{v}' + \delta \mathbf{e}_{s'}. \tag{1.149}$$

Then

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

Proof. Rewriting the hypothesis of the proposition as

$$\mathbf{r}_{d''} + \lambda \mathbf{P}_{d''}\mathbf{v}' = \mathbf{r}_{d'} + \lambda \mathbf{P}_{d'}\mathbf{v}' + \delta \mathbf{e}_{s'} = \mathbf{v}' + \delta \mathbf{e}_{s'}, \tag{1.150}$$

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

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

we have

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

Applying Theorem 1.2, which states

$$\mathbf{v}_{\lambda}^{(d'')^{\infty}} = (\mathbf{I} - \lambda \mathbf{P}_{d''})^{-1}\mathbf{r}_{d''},$$

the result follows.

□

The above proposition guarantees that if inequality (1.149) 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.

With this result we can easily prove that policy iteration converges. We refer to the vector-matrix description of PIA.

Theorem 1.16. 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 1.7 at each iteration, either $d'' = d'$ so that

$$\mathbf{v}' = \mathbf{r}_{d'} + \lambda \mathbf{P}_{d'} \mathbf{v}' = \mathbf{r}_{d''} + \lambda \mathbf{P}_{d''} \mathbf{v}' = \underset{d \in D^{\text{MD}}}{\text{c-max}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}'\} = L\mathbf{v}' \tag{1.151}$$

or there is a strict improvement in at least one state.

Since there are only a finite number of stationary deterministic policies and since no policy can repeat prior to termination as in (1.151), PIA terminates with a solution of the Bellman equation $\mathbf{v} = L\mathbf{v}$. □

Some comments follow:

1. As a consequence of the remarks in the preceding section and the above proof, policy iteration converges in a finite state and action model if updates occur immediately after a identifying a strict improvement in some state.

2. Theorem 1.16 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. In the next section, we provide a general approach for analyzing policy iteration.

1.6.3 Policy Iteration and Newton's Method*

It is now well appreciated that policy iteration may be viewed as equivalent to 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 convergence rate. In this section, we discuss this equivalence.

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

$$x' = x - \frac{f(x)}{f'(x)},$$

where the prime indicates derivative. When $f(x)$ is a differentiable, convex function and $|f'(x) - f'(y)| \leq K|x - y|$ for some $K > 0$, Newton's method converges quadratically.[12]

[12]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^*.

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

$$\mathbf{x}' = \mathbf{x} - (J(\mathbf{x}))^{-1}F(\mathbf{x}) \tag{1.152}$$

where $J(\mathbf{x})$ denotes the *Jacobian matrix*[13] of $F(\cdot)$ evaluated at $\mathbf{x}$.

To illustrate how Newton's method arises in a Markov decision process, we recast the problem of finding a fixed point of L as that of finding a *zero* of B, as defined in (1.110), where

$$B\mathbf{v} = L\mathbf{v} - \mathbf{v} = \underset{d \in D^{\text{MD}}}{\text{c-max}}\{\mathbf{r}_d + (\lambda \mathbf{P}_d - \mathbf{I})\mathbf{v}\} \tag{1.153}$$

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

From Lemma 1.5, for $\mathbf{u} \in V$ and $\mathbf{v} \in V$,

$$B\mathbf{u} \geq B\mathbf{v} + (\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I})(\mathbf{u} - \mathbf{v}) \tag{1.154}$$

where $d_\mathbf{v}$ is a $\mathbf{v}$-improving 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 a property of differentiable, convex functions from $\Re^n$ to $\Re^n$ that

$$F(\mathbf{y}) \geq F(\mathbf{x}) + J(\mathbf{x})(\mathbf{y} - \mathbf{x}) \tag{1.155}$$

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

The following provides a closed form representation for the iterates of the PIA. Note that we also define the operator $Z : V \to V$, which we will use below.

Proposition 1.8. (**Newton method representation for policy iteration**) Let $\mathbf{v}$ and $\mathbf{v}'$ be successive iterates of the policy iteration algorithm. Then

$$\mathbf{v}' = \mathbf{v} - (\lambda \mathbf{P}_{d_\mathbf{v}} - \mathbf{I})^{-1} B\mathbf{v} := Z\mathbf{v} \tag{1.156}$$

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

Proof. Let $d_\mathbf{v}$ be selected in (1.145) in the improvement step when entered with $\mathbf{v}$ and let $\mathbf{v}'$ be obtained at the next evaluation step by solving (1.144). Then

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

□

[13]The *Jacobian matrix* of a function $F(\cdot) : \Re^n \to \Re^n$ 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.

Thus, referring to (1.152), we see 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, and so it does not have a unique "derivative" at its break points. However using $\lambda \mathbf{P}_{d_{\mathbf{v}}} - \mathbf{I}$ corresponding to any $\mathbf{v}$-improving decision rule can assume this role. Figure 1.8 below illustrates these concepts.

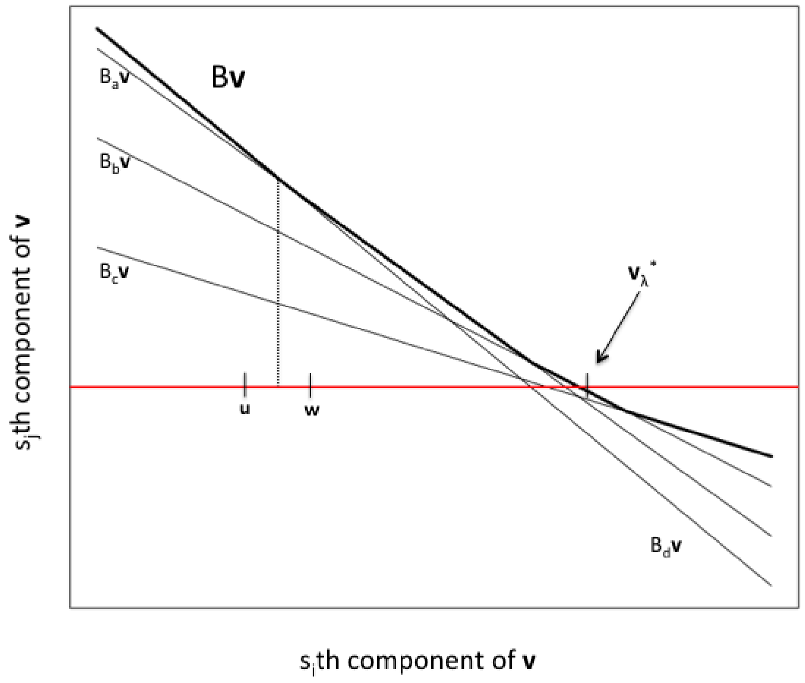


Figure 1.8: Bold line represents $B\mathbf{v}(s_j)$ as a function of $\mathbf{v}(s_i)$ for two states s_i and s_j. Note that it is piecewise linear and convex decreasing with respect to $\mathbf{v}(s_i)$. Each light line corresponds to a component of $B_{d'}\mathbf{v}(s_j) = \mathbf{r}_{d'}(s_j) + (\lambda \mathbf{P}_{d'} - \mathbf{I})\mathbf{v}(s_j)$ 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 component of the optimal value function when $B\mathbf{v}(s_j) = 0$. The quantities $\mathbf{w}$ and $\mathbf{u}$ are referred to below when discussing convergence rates of policy iteration.

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.

Lemma 1.6. 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 1.8 to show that $Z\mathbf{v}$ is the value of $(d_\mathbf{v})^\infty$.

The second result follows from rewriting (1.156) 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 we use the first result in Lemma 1.3 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 1.3 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 (1.154) 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 1.12.3.

1.7 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 we 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 appointment scheduling and multi-product inventory control where there are a large number of actions in each state.

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.

1.7.1 Motivation

We motivate MPI as follows. From Proposition 1.8, 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 1.2. 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

$$\mathbf{v}' = \mathbf{v} + \sum_{n=0}^{m} (\lambda \mathbf{P}_{d_\mathbf{v}})^n B\mathbf{v}. \tag{1.157}$$

Note that when $m = 0$, (1.157) 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. We refer to m as the *truncation order* of the algorithm.

Equation (1.157) is not efficient for calculation. Instead we implement it by executing one value iteration step $\mathbf{v}' = L\mathbf{v}$ to identify a $\mathbf{v}$-improving decision rule, $d_\mathbf{v}$ according to

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

and then apply value iteration m times for the *fixed* decision rule $d_\mathbf{v}$ as follows

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

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

Modified policy iteration: implementation recursion

$$\mathbf{v}' = L_{d_\mathbf{v}}^m L\mathbf{v} = L_{d_\mathbf{v}}^{m+1}\mathbf{v}. \tag{1.158}$$

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 $\mathbf{v}$.

We now show that these representations are equivalent. That is the truncated series representation can be evaluated by one step of value iteration followed by m steps of applying $L_{d_\mathbf{v}}$.

Proposition 1.9. For each $\mathbf{v} \in V$,

$$\mathbf{v} + \sum_{n=0}^{m} (\lambda \mathbf{P}_{d_\mathbf{v}})^n B\mathbf{v} = L_{d_\mathbf{v}}^m L\mathbf{v}. \tag{1.159}$$

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

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

□

1.7.2 A Modified Policy Iteration Algorithm

We first describe an MPI algorithm using vector notation and then an implementable form using component notation. Unlike policy iteration, MPI is not a finite algorithm, so we include a span-based stopping criterion. We choose it so that the algorithm terminates with an ϵ-optimal policy and an approximate value function that is within ϵ of the optimal value based on a lower bound extrapolation.

Modified policy iteration: vector notation

1. **Initialize:** Specify a decision rule $d' \in D^{\mathrm{MD}}$, $\mathbf{v} \in V$, a sequence of indices $\{m_n : n = 0, 1, \dots\}$ and a stopping criterion $\epsilon > 0$. Set $n = 0$.
2. While $\mathrm{sp}(L\mathbf{v} - \mathbf{v}) > \epsilon$:
   (a) **Evaluate:** Compute
   $$\mathbf{v}' \leftarrow L_{d'}^{m_n+1}\mathbf{v} \tag{1.160}$$
   (b) **Improve:** Compute $L\mathbf{v}$ and choose
   $$d' \in \underset{d \in D^{\mathrm{MD}}}{\arg \text{c-max}}\, L_d\mathbf{v}. \tag{1.161}$$
   (c) **Iterate:** Set $n \leftarrow n + 1$ and $\mathbf{v} \leftarrow \mathbf{v}'$.
3. **Approximate the optimal value function:** Set
$$\mathbf{v}_\lambda^\epsilon = \mathbf{v} + \lambda(1-\lambda)^{-1}(\underline{L\mathbf{v} - \mathbf{v}})\mathbf{e}.$$

We comment on the above algorithm.

1. In the initialization we specify both a decision rule and initial value function. This avoids calculating $L\mathbf{v}$ but is included to simplify the flow of the algorithm. Alternatively one could start with just a value $\mathbf{v}$ and compute $L\mathbf{v}$ and then choose d' according to (1.161). We could choose $\mathbf{v}$ such that $v(s) = (1-\lambda)^{-1} \min_{a \in A_s} r(s, a)$, which would be a lower bound on $\mathbf{v}_\lambda^*$. Choosing such an initial value guarantees monotone convergence.

2. One-advantage of starting with a decision rule is that you can choose it to have a specific structure that is appropriate for the model under consideration. In this case, if $\mathbf{v} = \mathbf{0}$, then the first evaluation, $L_{d'}^{m_0+1}\mathbf{0}$ represents the value of using decision rule d' for $m_0 + 1$ decision epochs.

3. Since $L\mathbf{v} = L_{d'}\mathbf{v}$, (1.158) shows that when computing $L_{d'}^{m_n+1}\mathbf{v}$ in the evaluation step, the first iteration has already been computed in the previous “Improve” step.

4. 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.

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

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

7. The evaluation steps can be implemented using Gauss-Seidel iteration.

We now present the above algorithm in component notation. We initialize it by specifying an action for each state. As noted above, we could begin with any $\mathbf{v} \in V$ but a lower bound on the optimal value function seems preferable. Moreover to find an improving action requires little additional effort beyond computing $v'(s)$. To reconcile this with vector notation, note that $v'(s) = L\mathbf{v}(s)$. For transparency, we denote the iterates of the evaluation step by $u(s)$ and $u'(s)$. The only reason for including the iteration index n is so that we can use the appropriate truncation order m_n at each iteration. The algorithm terminates with an ϵ-optimal policy $(d^\epsilon)^\infty$ and an approximation that is within ϵ (in norm) of the optimal value function.

Modified Policy Iteration: component notation

1. **Initialization:** For each $s \in S$, choose $a'_s \in A_s$ and set $v(s) = 0$. Specify a sequence $\{m_n : n = 0, 1\}$, $\epsilon > 0$ and set $n = 0$.
2. While $\text{sp}(\mathbf{v}' - \mathbf{v}) > \lambda(1-\lambda)^{-1}\epsilon$,
   (a) **Evaluation:**
      i. $m \leftarrow 1$ and $u(s) \leftarrow v(s)$ for all $s \in S$.
      ii. While $m \leq m_n$
         A. For each $s \in S$:
$$u'(s) = r(s, a'_s) + \lambda \sum_{j \in S} p(j|s, a'_s)u(j).$$
         B. $u(s) = u'(s)$ for all $s \in S$.
         C. $m \leftarrow m + 1$.
      iii. For all $s \in S$, $v(s) = u'(s)$.
   (b) **Improvement:**
      i. For each $s \in S$,
$$v'(s) = \max_{a \in A_s} \left\{ r(s, a) + \lambda \sum_{j \in S} p(j|s, a)v(j) \right\}.$$
      ii. For each $s \in S$, select
$$a'_s \in \arg\max_{a \in A_s} \left\{ r(s, a) + \lambda \sum_{j \in S} p(j|s, a)v(j) \right\}.$$
   (c) **Iterate:** Set $v(s) = v'(s)$ for all $s \in S$ and $n \leftarrow n + 1$.
3. **Termination:** For all $s \in S$, set
   (a) $d^{\epsilon}(s) = a'_s$
   (b) $v^{\epsilon}_{\lambda} = v'(s) + \lambda(1-\lambda)^{-1}(\underline{\mathbf{v}' - \mathbf{v}})$

Example

Example 1.15. We solve the two-state example using modified policy iteration with $\lambda = 0.9$ and $\epsilon = 0.000001$. For simplicity, we let $m_n = 3$ for all n.

1. Choose $a'_{s_1} = a_{1,2}$ and $a'_{s_2} = a_{2,1}$.
2. After three iterations of value iteration, we have $v(s_1) = -3.55$ and $v(s_2) = -13.55$.
3. In the improvement step we have

$$\begin{aligned} v'(s_1) &= \max_{i=1,2} \left\{ r(s_1, a_{1,i}) + \lambda \sum_{j=1,2} p(s_j|s_1, a_{1,i}) v(s_j) \right\} \\ &= \max\{3 + 0.9(0.8v(s_1) + 0.2v(s_2)), 5 + 0.9v(s_2)\} \\ &= \max\{-1.995, -7.195\} = -1.955, \end{aligned}$$

$$\begin{aligned} v'(s_2) &= \max_{i=1,2} \left\{ r(s_2, a_{2,i}) + \lambda \sum_{j=1,2} p(s_j|s_2, a_{2,i}) v(s_j) \right\} \\ &= \max\{-5 + 0.9v(s_2), 2 + 0.9(0.4v(s_1) + 0.6v(s_2))\} \\ &= \max\{-17.195, -6.595\} = -6.595, \end{aligned}$$

$$\begin{aligned} a'_{s_1} &= \arg\max_{i=1,2} \left\{ r(s_1, a_{1,i}) + \lambda \sum_{j=1,2} p(s_j|s_1, a_{1,i}) v(s_j) \right\} \\ &= \arg\max\{3 + 0.9(0.8v(s_1) + 0.2v(s_2)), 5 + 0.9v(s_2)\} \\ &= \arg\max\{-1.995, -7.195\} = a_{1,1}, \end{aligned}$$

and

$$\begin{aligned} a'_{s_2} &\in \arg\max_{i=1,2} \left\{ r(s_2, a_{2,i} + \lambda \sum_{j=1,2} p(s_j|s_2, a_{2,i}) v(s_j) \right\} \\ &\in \arg\max\{-5 + 0.9v(s_2), 2 + 0.9(0.4v(s_1) + 0.6v(s_2))\} \\ &\in \arg\max\{-17.195, -6.595\} = a_{2,2}. \end{aligned}$$

4. Since $\text{sp}(\mathbf{v}' - \mathbf{v}) = 5.4 \geq \lambda(1-\lambda)^{-1}\epsilon = 0.000009$, $\mathbf{v} \leftarrow \mathbf{v}'$ and we repeat.
5. We run four additional iterations of modified policy iteration, and the results of each iteration are summarized in Table 1.1.
6. Since $\text{sp}(\mathbf{v}' - \mathbf{v}) = 0.0000032 < \lambda(1-\lambda)^{-1}\epsilon = 0.000009$, we terminate with $d^\epsilon(s_1) = a_{1,2}$ and $d^\epsilon(s_2) = a_{2,2}$.

Iteration	$v(s_1)$	$v(s_2)$	$v'(s_1)$	$v'(s_2)$	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	$a_{1,2}$	$a_{2,2}$	0.0114
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 1.1: Modified policy iterations from Example 1.15.

1.7.3 Convergence of modified policy iteration*

Figure 1.9 illustrates how the iterates of modified policy iteration evolve and motivates the proof below. Note we suppress the component index in the following discussion. That is we write u^k in place of $\mathbf{u}^k(s_i)$ and Bu^k in place of $B\mathbf{u}^k(s_j)$. The quantities $\{u^0, \dots, u^5\}$ in the figure can be thought of in several ways.

1. In all cases iteration starts at u^0.
2. $u^1(s_i) = u^0 + Bu^0 = Lu^0$ and $u^2 = u^1 + Bu^1 = Lu^1$ may be regarded as the result of applying two iterates of value iteration or equivalently modified policy iteration of order 0.
3. u^1, u^2 and u^3 may be regarded as the result of applying one step of modified policy iteration or order 2. That is, the first step identifies the u^0-improving decision rule d and evaluates u^1 and the next two steps compute $u^2 = u^1 + B_d u^1 = L_d u^1$ and $u^2 = u^1 + B_d u^2 = L_d u^2$. Then the next full step of modified policy iteration of order 2 begins at u^3 by identifying a u^3-improving decision rule and computing u^4 and then uses this decision rule to evaluate u^5. This approach avoids two maximizations at each full MPI iteration.

We show that modified policy iteration converges using the same approach that was suggested for establishing the convergence of policy iteration in a non-finite setting. To do so, we define the operator $W_m : V \to V$ to represent the right hand side of (1.157), 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 1.6 can be easily modified to establish the analogous result for W_m. Only statement 4 requires a revised argument.

Lemma 1.7. Suppose for some $\mathbf{v} \in V$ that $B\mathbf{v} \geq \mathbf{0}$ then for any integer $m \geq 0$,

1. $\mathbf{v}^*_\lambda \geq W_m \mathbf{v}$,

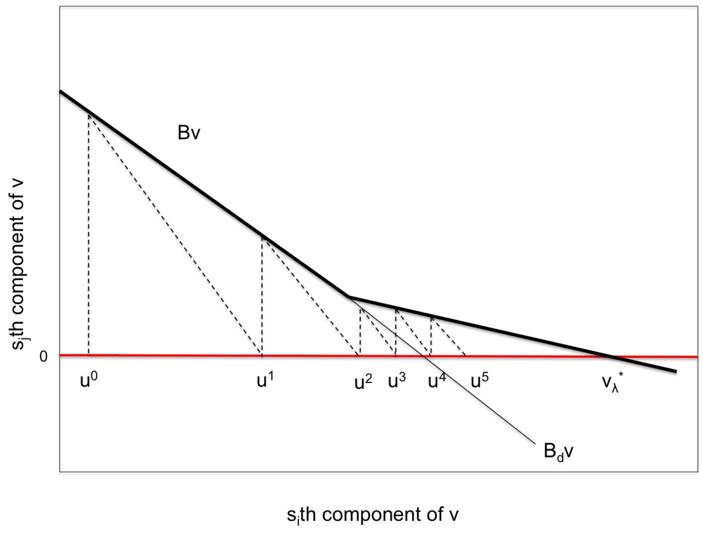


Figure 1.9: Illustration of modified policy iteration updates. The bold line represents $B\mathbf{v}(s_j)$ as a function of $\mathbf{v}(s_i)$ for two states s_i and s_j and the lighter line represents $B_d\mathbf{v}(s_j)$ as a function of $\mathbf{v}(s_i)$ for the u^0-improving decision rule d. Note that u^3 is apparently computed in a different way than the other updates.

2. $W_m\mathbf{v} \geq L\mathbf{v}$,
3. $W_m\mathbf{v} \geq \mathbf{v}$, and
4. $B(W_m\mathbf{v}) \geq \mathbf{0}$.

Theorem 1.17. Let $(m_n)_{n=0}^{\infty}$ denote a sequence of non-negative integers and let $(\mathbf{v}^n)_{n=0}^{\infty}$ denote a sequence of iterates $\mathbf{v}^n \in V$ generated by 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 $\mathbf{u}^0 = \mathbf{v}^0$. Define the sequence $\mathbf{u}^{n+1} = L\mathbf{u}^n$ and as noted above $\mathbf{v}^{n+1} = W_{m_n}\mathbf{v}^n$. We show by induction that for all n, $B\mathbf{v}^n \geq \mathbf{0}$ and

$$\mathbf{u}^n \leq \mathbf{v}^n \leq \mathbf{v}_\lambda^*. \tag{1.162}$$

Recall that $B\mathbf{v} \geq \mathbf{0}$ is equivalent to $L\mathbf{v} \geq \mathbf{v}$. Hence $L^2\mathbf{v}^0 \geq L\mathbf{v}^0 \geq \mathbf{v}^0$ 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 (1.162) and $B\mathbf{v}^n \geq \mathbf{0}$ hold for all $m \leq n$. By the second result in Lemma 1.7 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 1.7 and that $B\mathbf{v}^{n+1} \geq \mathbf{0}$ follows from the fourth result of Lemma 1.7. That $\mathbf{v}^{n+1} \geq \mathbf{v}^n$ is an application of the third result of Lemma 1.7. □

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. Example 6.5.1 in Puterman [1994] shows that this conjecture is false.

The following more general convergence result was established by Canbolat and Rothblum [2013].

Theorem 1.18. For any starting value, modified policy iteration converges monotonically and in a finite number of iterations to an ϵ-optimal policy and value function.

1.8 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 **??**). In this section, we describe the relationship between linear programs (LPs) and MDPs and show that key MDP concepts have direct analogues in linear programming. This section requires background on linear algebra and linear programming, some of which appears in Appendix **??**.

1.8.1 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

$$v(s) \geq \; r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v(j) \tag{1.163}$$

for all $a \in A_s$ and $s \in S$. By Corollary 1.2, any $v(s)$ satisfying (1.163) 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 (1.163) for all $a \in A_s$ and $s \in S$. To find such a $v(s)$, we choose $\alpha(s) > 0$ for all $s \in S$ and solve the following linear program.

Primal LP formulation of a discounted MDP

$$\text{minimize} \quad \sum_{s \in S} \alpha(s)v(s) \tag{1.164a}$$

$$\text{subject to} \quad v(s) - \lambda \sum_{j \in S} p(j|s,a)v(j) \geq r(s,a), \quad a \in A_s, s \in S \tag{1.164b}$$

Some comments about this formulation follow:

1. We refer to the above model as the *primal* linear program. Associated with every primal linear is a *dual* linear program, which we describe below. We note that the relationship between policies and values becomes more apparent by analyzing the dual linear program.

2. We emphasize that $v(s)$ is *unrestricted* in sign for all $s \in S$. That is, it is not restricted to be non-negative, for example.

3. This LP has $|S|$ variables and $\sum_{s \in S} |A_s|$ constraints. Thus, the number of constraints could far exceed the number of variables.

4. Choosing $\alpha(s)$ so that $\sum_{s \in S} \alpha(s) = 1$ allows us to interpret this quantity as a probability distribution on S.

5. Note that there is exactly one positive coefficient in each row[14], the coefficient of $v(s)$, and its value is $1 - \lambda p(s|s,a)$.

6. Since

Example 1.16. Consider Example **??**. The Bellman equations are given in equations (1.82) and (1.83). Since the problem has two states, we only need to define a single parameter α assuming the objective function coefficients sum to 1. The primal linear program is

$$\begin{aligned} \text{minimize} \quad & \alpha v(s_1) + (1-\alpha)v(s_2) \\ \text{subject to} \quad & (1-0.8\lambda)v(s_1) - 0.2\lambda v(s_2) \geq 3 \\ & 1v(s_1) - \lambda v(s_2) \geq 5 \\ & 0v(s_1) + (1-\lambda)v(s_2) \geq -5 \\ & (1-0.4\lambda)v(s_1) - 0.6\lambda v(s_2) \geq 2. \end{aligned} \tag{1.165}$$

The four constraints correspond respectively to choosing actions $a_{1,1}$ or $a_{1,2}$ in

[14] A linear system with this property is referred to as *Leontief substitution system*.

s_1 and $a_{2,1}$ or $a_{2,2}$ in s_2. For $\lambda = 0.9$, the four constraints can be written as:

$$\begin{aligned} v(s_2) &\leq \frac{14}{9}v(s_1) - \frac{50}{3} \\ v(s_2) &\leq \frac{10}{9}v(s_1) - \frac{50}{9} \\ v(s_2) &\geq -50 \\ v(s_2) &\geq \frac{18}{23}v(s_1) + \frac{100}{23}. \end{aligned}$$

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 we want 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 1.11. The figure below illustrates the region described by the above constraints and illustrates a graphical solution for $\lambda = 0.9$. Note that we can obtain the optimal policy by looking at which constraints are tight at the optimal solution. The second constraint corresponds 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, notice that the optimal solution corresponds to the "lower left" vertex of the feasible region. So any value of $\alpha \in [0, 1]$ will result in the same optimal solution, as we will show formally in Corollary 1.5.

LP formulation: vector notation

We now provide a vector formulation of the above linear program. We find this representation useful for establishing some theoretical results below. Using the same logic as above, if $\mathbf{v} \in V$ is a solution of the Bellman equation

$$\mathbf{v} = \underset{d \in D^{\text{MD}}}{\text{c-max}}\{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\} = L\mathbf{v}, \tag{1.166}$$

then it satisfies

$$\mathbf{v} \geq \mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v} \tag{1.167}$$

for all $d \in D^{\text{MD}}$. Proceeding as above, let $\boldsymbol{\alpha} > \mathbf{0}$ be a given vector of length $|S|$. This leads to the following linear program.

Primal LP formulation of a discounted MDP: vector notation

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{v} && (1.168\text{a}) \\ \text{subject to} \quad & (\mathbf{I} - \lambda \mathbf{P}_d)\mathbf{v} \geq \mathbf{r}_d, \quad d \in D^{\text{MD}}. && (1.168\text{b}) \end{aligned}$$

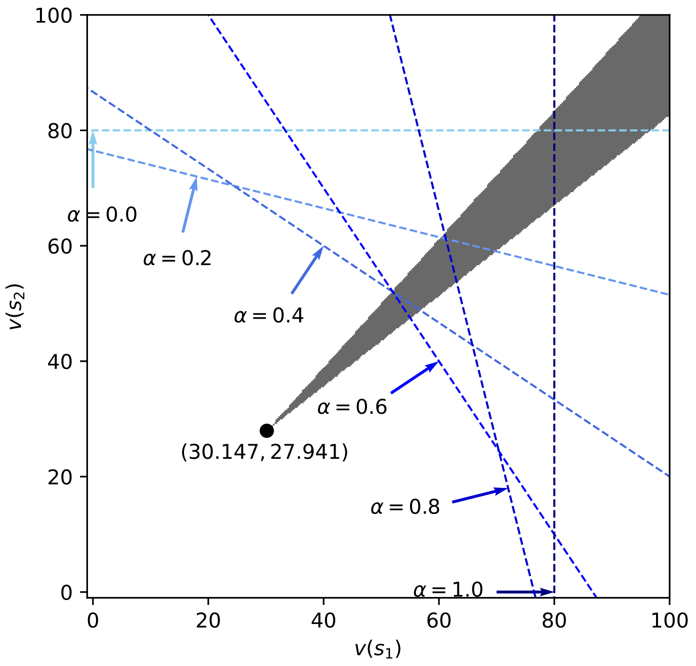


Figure 1.10: Figure for Example 1.16. Shaded region represents the feasible region, defined by the constraints. Different α values represent different objective functions. The identified vertex is the optimal solution for all α.

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 $d_2 = \{a_{1,1}, a_{2,2}\}$ from Example **??**. 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. Notice that the redundancy arises in going from (1.166) to (1.167), where the component-wise maximum over decision rules (i.e., a maximum over actions) was replaced with an enumeration over decision rules.

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 the MDP has exactly one positive coefficient in each row.

Theorem 1.19. Let $\boldsymbol{\alpha} > \mathbf{0}$ and assume $0 \leq \lambda < 1$.

1. The linear program (1.164) 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
$$v^*(s) - \lambda \sum_{j \in S} p(j|s, a_s^*) v^*(j) = r(s, a_s^*). \tag{1.169}$$
3. For each $s \in S$, let $d^*(s) = a_s^*$ as defined in (1.169). 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 (1.164b) 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 $v'(s)$ is a feasible solution of the linear program. Note that for any $d \in D^{\text{MD}}$,
$$\mathbf{v}_\lambda^{d^\infty} = (\mathbf{I} - \lambda \mathbf{P}_d)^{-1} \mathbf{r}_d \leq (\mathbf{I} - \lambda \mathbf{P}_d)^{-1} R\mathbf{e} = (1-\lambda)^{-1} R\mathbf{e} = \mathbf{v}', \tag{1.170}$$
where the first equality is due to Theorem 1.2, the inequality is due to the second result of Lemma 1.3, and the second equality is due to the third result of Lemma 1.3. So, the objective function value of $\mathbf{v}'$ is bounded below by
$$\max_{d \in D^{\text{MD}}} \sum_{s \in S} \alpha(s) v_\lambda^{d^\infty}(s). \tag{1.171}$$
Hence by Theorem **??** in Appendix **??**, the linear program has an optimal solution, which we denote 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 $a \in A_{s'}$
$$v^*(s') - \lambda \sum_{j \in S} p(j|s',a) v^*(j) > r(s',a). \tag{1.172}$$
Hence there exists an $\epsilon > 0$ for which
$$v^*(s') - \lambda \sum_{j \in S} p(j|s',a) v^*(j) - \epsilon \geq r(s',a).$$
Notice that
$$(v^*(s') - \epsilon) - \lambda \left(\sum_{j \in S \setminus \{s'\}} p(j|s',a) v^*(j) + p(s'|s',a)(v^*(s') - \epsilon) \right) \geq v^*(s') - \lambda \sum_{j \in S} p(j|s',a) v^*(j) - \epsilon$$

since $\lambda p(s'|s', a)\epsilon > 0$. 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 1.2 imply that $v^*(s) = v_\lambda^{(d^*)^\infty}(s)$ where $d^*(s) = a_s^*$ for all $s \in S$. By Theorem 1.7, 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 1.5) 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

$$v^*(s) - \lambda \sum_{j \in S} p(j|s, a) v^*(j) \geq r(s, a) \tag{1.173}$$

holds with equality. Hence solving the LP and noting which constraints hold with equality identifies an optimal policy. When two or more actions satisfy (1.173) 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 1.19 remains valid for any $\boldsymbol{\alpha} > \mathbf{0}$, yielding the following important result.

Corollary 1.5. 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 (1.164b). Furthermore, the feasible region must be contained in the region

$$\{\mathbf{v} \in V | \mathbf{v} = \mathbf{v}^* + \mathbf{y}, \mathbf{y} \geq \mathbf{0}\}. \tag{1.174}$$

1.8.2 The Dual Linear Program

By analyzing the Markov decision process through its dual linear programming formulation, we obtain a direct way of identifying optimal policies and insight into the relationship between between feasible solutions and stationary policies. The key result of this section is given in Theorem 1.24, where we show 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 **??**), the dual linear program for the discounted MDP may be expressed in as follows.

Dual LP formulation of discounted MDP

$$\text{maximize} \quad \sum_{s\in S}\sum_{a\in A_s} r(s,a)x(s,a) \tag{1.175a}$$

$$\text{subject to} \quad \sum_{a\in A_s} x(s,a) - \sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a)x(j,a) = \alpha(s), \quad s\in S \tag{1.175b}$$

$$x(s,a) \geq 0, \quad a\in A_s, s\in S \tag{1.175c}$$

Some comments on 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 the constraints and $r(s,a)$ appears in the objective function.

2. In contrast to the primal, the objective function of the dual is maximization, instead of minimization.

3. The dual linear program has $|S|$ rows and $\sum_{s\in S}|A_s|$ variables. Hence there are many more columns than rows.

4. In contrast to the primal, the dual variables are constrained to be non-negative.

5. When we seek a numerical solution, we usually use the dual formulation.

6. Note that j represents the starting state and s represents the state reached after one transition in equation (1.175b), reflecting the fact that the dual is written using the transpose of the transition probability matrix.

Example 1.17. The dual linear program of (1.165) can be written as follows:

$$\begin{aligned}
\text{minimize} \quad & 3x(s_1,a_{1,1}) + 5x(s_1,a_{1,2}) - 5x(s_2,a_{2,1}) + 2x(s_2,a_{2,2}) \\
\text{subject to} \quad & x(s_1,a_{1,1}) + x(s_1,a_{1,2}) - 0.8\lambda x(s_1,a_{1,1}) - 0.4\lambda x(s_2,a_{2,2}) = \alpha \\
& x(s_2,a_{2,1}) + x(s_2,a_{2,2}) - 0.2\lambda x(s_1,a_{1,1}) \qquad (1.176)\\
& -\lambda x(s_1,a_{1,2}) - \lambda x(s_2,a_{2,1}) - 0.6\lambda x(s_2,a_{2,2}) = 1-\alpha \\
& x(s_1,a_{1,1}), x(s_1,a_{1,2}), x(s_2,a_{2,1}), x(s_2,a_{2,2}) \geq 0.
\end{aligned}$$

Letting $\lambda = 0.9$ and $\alpha = 0.5$, we get an optimal solution of $x^*(s_1,a_{1,1}) = 0, x^*(s_1,a_{1,2}) = 3.0147, x^*(s_2,a_{2,1}) = 0$ and $x^*(s_2,a_{2,2}) = 6.9853$.

Recall from Example 1.16 that $v^*(s_1) = 30.147$ and $v^*(s_2) = 27.941$. Thus, $\sum_s \alpha(s)v^*(s) = 29.044 = \sum_s \sum_{a\in A_s} x^*(s,a)r(s,a)$. This result, where the primal and dual optimal solutions have the same objective function value, is known as Strong Duality (see Theorem 1.23).

Where convenient, we will refer to the dual solution, $x(s,a), a \in A_s, s \in S$ as the vector $\mathbf{x}$. One may think of $\mathbf{x}$ as a "long" vector, where the first $|S|$ components correspond to $x(s,a_1), s \in S$, the second $|S|$ components correspond to, $x(s,a_2), s \in S$, and so on.

1.8.3 Dual feasible solutions and stationary policies

We show that there is a direct relationship between feasible solutions to the dual problem and stationary policies. First, we define two quantities. For each $d \in D^{\mathrm{MR}}, a \in A_s$ and $s \in S$, define $x_d(s,a)$ by:

$$x_d(s,a) := \sum_{j\in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} p^{d^\infty}(X_n = s, Y_n = a | X_1 = j). \qquad (1.177)$$

We interpret $x_d(s,a)$ as the total discounted joint probability that the MDP is in state s and chooses action a over the infinite horizon under policy d^∞ when the process starts in state $j \in S$ with probability $\alpha(j)$.

Before showing how to construct a randomized decision rule from a feasible solution to the dual linear program, we need the following lemma.

Lemma 1.8. Let $x(s,a)$ be a feasible solution to the dual linear program. Then $\sum_{a\in A_s} x(s,a) > 0$.

Proof. From (1.175b) and the condition that $x(s,a) \geq 0$ for all $a \in A_s, s \in S$ it follows 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 problem, we define the Markovian randomized decision rule $d_{\mathbf{x}}$ to be that which chooses action a in state s with probability given by:

$$w_{d_{\mathbf{x}}}(s,a) := \frac{x(s,a)}{\sum_{a' \in A_s} x(s,a')}. \tag{1.178}$$

Lemma 1.8 implies that $w_{d_{\mathbf{x}}}(s,a)$ is well defined.

The relationship is quite subtle. If we start with a feasible $\mathbf{x}$, define $d_{\mathbf{x}}(s)$ by (1.178) and then compute $x_{d_{\mathbf{x}}}(s,a), a \in A_s, s \in S$ by (1.177), we get back to our original $\mathbf{x}$. The following theorem relates $\mathbf{x}$ and $d_{\mathbf{x}}(s)$. Its proof is straightforward but intricate.

Theorem 1.20. Suppose $0 \leq \lambda < 1$ and $\alpha(s) > 0$ for all $s \in S$. Then

1. For any $d \in D^{\mathrm{MR}}$, $x_d(s,a), a \in A_s, s \in S$ is a feasible solution to the dual LP.
2. For any feasible solution $\mathbf{x}$ to the dual LP, $x_{d_{\mathbf{x}}}(s,a) = x(s,a)$ for all $a \in A_s$ and $s \in S$.

Proof. By definition $x_d(s,a) \geq 0$ for all $a \in A_s$, $s \in S$. We write the expression on the left hand side of the equality constraint (1.175b) as

$$\sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a) x_d(j,a) = \sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a) \sum_{k\in S}\alpha(k)\sum_{n=1}^{\infty}\lambda^{n-1}p^{d^\infty}(X_n = j, Y_n = a|X_1 = k) \tag{1.179}$$

$$= \sum_{k\in S}\alpha(k)\left(\sum_{n=1}^{\infty}\lambda^{n}p^{d^\infty}(X_{n+1} = s|X_1 = k)\right) \tag{1.180}$$

$$= \sum_{k\in S}\alpha(k)\left(\sum_{n=1}^{\infty}\lambda^{n-1}p^{d^\infty}(X_n = s|X_1 = k) - p^{d^\infty}(X_1 = s|X_1 = k)\right) \tag{1.181}$$

$$= \sum_{a\in A_s}\sum_{k\in S}\alpha(k)\sum_{n=1}^{\infty}\lambda^{n-1}p^{d^\infty}(X_n = s, Y_n = a|X_1 = k) - \alpha(s) \tag{1.182}$$

$$= \sum_{a\in A_s} x_d(s,a) - \alpha(s). \tag{1.183}$$

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. Define

$$u(s) := \sum_{a\in A_s} x(s,a), \tag{1.184}$$

which by Lemma 1.8 is strictly positive for each $s \in S$. Rewriting the equality constraint in the dual,

$$\begin{aligned}
\alpha(s) &= u(s) - \sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a)x(j,a) &(1.185)\\
&= u(s) - \sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a)x(j,a)\frac{u(j)}{\sum_{a\in A_j} x(j,a)} &(1.186)\\
&= u(s) - \sum_{j\in S}\sum_{a\in A_j} \lambda p(s|j,a)w_{d_{\mathbf{x}}}(j,a)u(j) &(1.187)\\
&= u(s) - \sum_{j\in S} \lambda P_{d_{\mathbf{x}}}(s|j)u(j), &(1.188)
\end{aligned}$$

where $P_{d_{\mathbf{x}}}(s|j)$ is defined in (1.3). Writing the last expression in matrix notation yields

$$\boldsymbol{\alpha}^{\mathsf{T}} = \mathbf{u}^{\mathsf{T}}(\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{x}}}). \tag{1.189}$$

From Lemma 1.2

$$\mathbf{u}^{\mathsf{T}} = \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{x}}})^{-1} = \boldsymbol{\alpha}^{\mathsf{T}}\left(\sum_{n=1}^{\infty} \lambda^{n-1}\mathbf{P}_{d_{\mathbf{x}}}^{n-1}\right). \tag{1.190}$$

writing (1.190) in component notation gives

$$\begin{aligned}
u(s) &= \sum_{j\in S}\alpha(j)\sum_{n=1}^{\infty}\lambda^{n-1}P_{d_{\mathbf{x}}}^{n-1}(s|j)\\
&= \sum_{j\in S}\alpha(j)\sum_{n=1}^{\infty}\lambda^{n-1}\sum_{a\in A_s} p^{d_{\mathbf{x}}^{\infty}}(X_n = s, Y_n = a|X_1 = j) &(1.191)\\
&= \sum_{a\in A_s} x_{d_{\mathbf{x}}}(s,a) &(1.192)
\end{aligned}$$

for all $s \in S$, where (1.191) is a consequence of the law of total probability applied as follows:

$$P_{d_{\mathbf{x}}}^{n-1}(s|j) = p^{d_{\mathbf{x}}^{\infty}}(X_n = s|X_1 = j) = \sum_{a\in A_s} p^{d_{\mathbf{x}}^{\infty}}(X_n = s, Y_n = a|X_1 = j). \tag{1.193}$$

This establishes that

$$\sum_{a\in A_s} x(s,a) = \sum_{a\in A_s} x_{d_{\mathbf{x}}}(s,a). \tag{1.194}$$

To complete the proof, from the definition of $x_{d_{\mathbf{x}}}(s,a)$, for all $s \in S$ and $a \in A_s$,

$$\begin{aligned}
x_{d_{\mathbf{x}}}(s,a) &= \sum_{j\in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} p^{d_{\mathbf{x}}^{\infty}}(X_n = s, Y_n = a | X_1 = j) \\
&= \sum_{j\in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} p^{d_{\mathbf{x}}^{\infty}}(X_n = s | X_1 = j) w_{d_{\mathbf{x}}}(s,a) \\
&= \sum_{a\in A_s} x_{d_{\mathbf{x}}}(s,a) w_{d_{\mathbf{x}}}(s,a) && (1.195) \\
&= \sum_{a\in A_s} x_{d_{\mathbf{x}}}(s,a) \frac{x(s,a)}{\sum_{a'\in A_s} x(s,a')} && (1.196) \\
&= x(s,a). && (1.197)
\end{aligned}$$

where the third equality (1.195) is a consequence of (1.192) and the last equality is due to (1.194). □

Theorem 1.20 establishes that any stationary randomized policy generates a feasible solution to the dual problem, and that all feasible solutions to the dual problem are generated by stationary randomized policies.

We can further specialize this relationship by showing a correspondence between *basic* feasible solutions [15] to the dual problem and stationary *deterministic* policies.

Theorem 1.21. Suppose $0 \le \lambda < 1$ and $\alpha(s) > 0$ for all $s \in S$. Then

1. Let $d \in D^{\mathrm{MD}}$. Then $x_d(s,a), a \in A_s, s \in S$ is a basic feasible solution to the dual LP.
2. Let $\mathbf{x}$ be a basic feasible solution to the dual LP. Then $d_{\mathbf{x}} \in D^{\mathrm{MD}}$.

Proof. If $d \in D^{\mathrm{MD}}$, it follows from Theorem 1.20 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 1.8,

$$\sum_{a\in A_s} y(s,a) > 0 \text{ and } \sum_{a\in A_s} z(s,a) > 0$$

[15] 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.

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 (1.178) chooses one action in A_s with probability 1, so that it is deterministic. $\square$

The results presented in Theorems 1.20 and 1.21 can be succinctly represented in Figure 1.11. The polyhedron[16] 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 are a weighted average of deterministic policies.

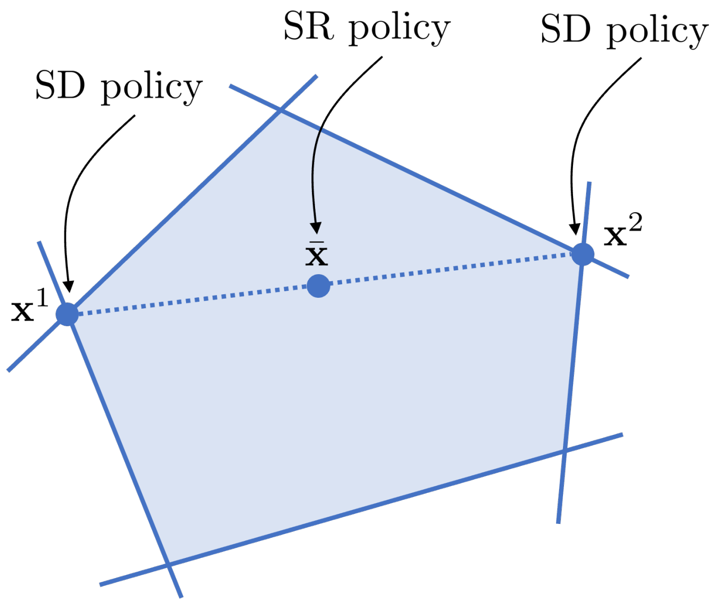


Figure 1.11: Relationship between feasible solutions of the dual LP and stationary policies of the MDP. Basic feasible solutions $\mathbf{x}^1$ and $\mathbf{x}^2$ correspond to stationary deterministic policies. Their convex combination, $\bar{\mathbf{x}}$, corresponds to a randomized policy.

To make things concrete, consider the two vertices $\mathbf{x}^1$ and $\mathbf{x}^2$ shown in Figure 1.11. They correspond to Markovian deterministic decision rules d_1 and d_2 that choose

[16]A *polyhedron* is a region formed by the intersection of a finite number of linear inequalities. See Appendix **??** for definitions of additional linear programming concepts.

actions $d_1(s)$ and $d_2(s)$, respectively, in state s. Now, consider a feasible solution $\bar{\mathbf{x}} = \mu\mathbf{x}^1 + (1-\mu)\mathbf{x}^2$, where $\mu \in (0,1)$, with corresponding Markovian randomized decision rule $d_{\bar{\mathbf{x}}}$. From equation (1.178), we have

$$w_{d_{\bar{\mathbf{x}}}}(s,a) = \frac{\bar{x}(s,a)}{\sum_{a'\in A_s}\bar{x}(s,a')} \tag{1.198}$$

$$= \frac{\mu x^1(s,a) + (1-\mu)x^2(s,a)}{\sum_{a'\in A_s}(\mu x^1(s,a') + (1-\mu)x^2(s,a'))} \tag{1.199}$$

By definition, for $i = 1,2$

$$x^i(s,a) = \begin{cases} \sum_{j\in S}\alpha(j)\sum_{n=1}^{\infty}\lambda^{n-1}p^{d_i^\infty}(X_n = s|X_1 = j) & \text{if } a = d_i(s) \\ 0 & \text{if } a \neq d_i(s). \end{cases} \tag{1.200}$$

Hence

$$w_{d_{\bar{\mathbf{x}}}}(s,a) = \begin{cases} \frac{\mu x^1(s,d_1(s))}{\mu x^1(s,d_1(s)) + (1-\mu)x^2(s,d_2(s))} & \text{if } a = d_1(s) \\ \frac{(1-\mu)x^2(s,d_2(s))}{\mu x^1(s,d_1(s)) + (1-\mu)x^2(s,d_2(s))} & \text{if } a = d_2(s) \\ 0 & \text{otherwise.} \end{cases} \tag{1.201}$$

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. We leave the proof as an exercise. This result allows us to provide a first principles proof of the existence of solutions to the dual LP.

Theorem 1.22. The feasible region of the dual LP is bounded.

As a preview of the results in the next section, we establish a connection between primal solutions (value functions) and dual solutions (discounted joint probability distributions).

Proposition 1.10. Let $\mathbf{x}$ be a dual feasible solution and $d_{\mathbf{x}} \in D^{\mathrm{MR}}$ be a randomized policy defined by (1.178). Then,

$$\sum_{s\in S}\sum_{a\in A_s} x_{d_{\mathbf{x}}^\infty}(s,a)r(s,a) = \sum_{j\in S}\alpha(j)v_\lambda^{d_{\mathbf{x}}^\infty}(j). \tag{1.202}$$

Proof. From equations (1.14) and (1.177), we have

$$v_\lambda^{d_{\mathbf{x}}^\infty}(j) = \sum_{s\in S}\sum_{a\in A_s}\sum_{n=1}^{\infty}\lambda^{n-1}p^{d_{\mathbf{x}}^\infty}(X_n = s, Y_n = a|X_1 = j)r(s,a), \tag{1.203}$$

and

$$x_{d_{\mathbf{x}}^\infty}(s,a) = \sum_{j \in S} \alpha(j) \sum_{n=1}^{\infty} \lambda^{n-1} p^{d\mathbf{x}^\infty}(X_n = s, Y_n = a | X_1 = j), \tag{1.204}$$

respectively. Thus,

$$\sum_{j \in S} \alpha(j) v_\lambda^{d_{\mathbf{x}}^\infty}(j) = \sum_{s \in S} \sum_{a \in A_s} x_{d_{\mathbf{x}}^\infty}(s,a) r(s,a). \tag{1.205}$$

□

The objective functions for both the primal and dual LPs can be interpreted as the expected total discounted reward of policy d^∞, but where the expectation is taken over the initial state distribution, in addition to that which results from randomized decision rules and probabilistic state transitions. Note the above result does not specify 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.

1.8.4 Optimal dual solutions and optimal policies

Theorem 1.23. 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

$$\sum_{s \in S} \alpha(s) v^*(s) = \sum_{s \in S} \sum_{a \in A_s} x^*(s,a) r(s,a). \tag{1.206}$$

Proof. The first result follows from Theorem 1.22 and Theorem **??**. The second result follows from Proposition 1.10. □

Note that Theorem 1.23 is simply a restatement of Strong Duality of linear programming (Theorem **??** in Appendix **??**) in our MDP context, and follows since we established in Theorem 1.19 that the primal LP has an optimal solution.

Next, we formalize the connection between optimal solutions to the dual LP and optimal policies of the MDP. This is the main result of this section.

Theorem 1.24. 1. Suppose $\mathbf{x}^*$ is an optimal solution to the dual LP. Then $d_{\mathbf{x}^*}^\infty$ 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 defined as in equation (1.178). Combining equation (1.205) and (1.206), we have

$$\sum_{s\in S}\alpha(s)v_\lambda^{(d_{\mathbf{x}^*})^\infty}(s)=\sum_{s\in S}\sum_{a\in A_s}x^*(s,a)r(s,a)=\sum_{s\in S}\alpha(s)v^*(s). \tag{1.207}$$

For all s, since $\alpha(s)>0$, it must be that

$$v_\lambda^{(d_{\mathbf{x}^*})^\infty}(s)=v^*(s). \tag{1.208}$$

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 $d\in D^{\text{MD}}$, or equivalently, $\mathbf{v}^*\geq L\mathbf{v}^*$. By Theorem 1.7, $\mathbf{v}^*\geq\mathbf{v}_\lambda^*$. Hence,

$$v_\lambda^{(d_{\mathbf{x}^*})^\infty}(s)\geq v_\lambda^*(s) \tag{1.209}$$

for all s. By optimality of $v_\lambda^*(s)$, $(d_{\mathbf{x}^*})^\infty$ is an optimal policy.

To prove the second result, from equation (1.205), we have

$$\begin{aligned}\sum_{s\in S}\sum_{a\in A_s}x_{d^*}(s,a)r(s,a)&=\sum_{s\in S}\alpha(s)v_\lambda^{(d^*)^\infty}(s) &(1.210)\\ &\geq\sum_{s\in S}\alpha(s)v_\lambda^{d^\infty}(s) &(1.211)\\ &=\sum_{s\in S}\sum_{a\in A_s}x_d(s,a)r(s,a) &(1.212)\\ &=\sum_{s\in S}\sum_{a\in A_s}x(s,a)r(s,a), &(1.213)\end{aligned}$$

for all $d\in D^{\text{MR}}$. Since any feasible solution $\mathbf{x}$ to the dual is associated with a decision rule $d\in D^{\text{MR}}$ such that $\mathbf{x}_d=\mathbf{x}$, no feasible solution has objective function value strictly greater than $\mathbf{x}_{d^*}$. Hence, $\mathbf{x}_{d^*}$ is optimal to the dual LP. □

The following corollary is a direct specialization of Theorem 1.24 to the case of basic feasible solutions, as consequence of Theorem 1.21.

Corollary 1.6.
1. Suppose $\mathbf{x}^*$ is an optimal basic feasible solution to the dual LP. Then $d_{\mathbf{x}^*}^\infty$ 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.

Using the dual LP, we can provide an alternative proof of the existence of a stationary deterministic optimal policy to the MDP. The feasible region includes non-negativity constraints, which means it does not contain a line[17], and thus has at least

[17]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 $\theta\in\Re$.

one vertex (see Theorem **??**). One of the fundamental theorems in linear programming is 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 be an optimal solution. Since a vertex corresponds to a stationary deterministic policy, this argument establishes that there will always exist an stationary optimal policy to the MDP that is deterministic.

Finally, recall that we showed previously that an optimal policy is optimal for all $\boldsymbol{\alpha} > \mathbf{0}$, using the primal LP. We can also establish this result using the dual.

Theorem 1.25. Let $\mathbf{x}^*$ be an optimal dual solution. The corresponding policy $d_{\mathbf{x}^*}^\infty$ is optimal for all $\boldsymbol{\alpha} > \mathbf{0}$.

Proof. First, we prove the result in 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 $a \neq d_{\mathbf{x}^*}(s)$, then $\mathbf{x}^*$ satisfies (1.175b):

$$(\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{x}^*}})^\mathsf{T} \mathbf{x}^* = \boldsymbol{\alpha}. \tag{1.214}$$

Since $\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{x}^*}}$ is invertible and since the transpose of the inverse is the inverse of the transpose,

$$\mathbf{x}^* = (\mathbf{I} - \lambda \mathbf{P}_{d_{\mathbf{x}^*}})^{-1} \boldsymbol{\alpha} \geq \mathbf{0}, \tag{1.215}$$

where the inequality follows from Lemma 1.3. 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 $d_{\mathbf{x}^*}^\infty$ 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 $\boldsymbol{\alpha} > \mathbf{0}$. □

1.8.5 Algorithms

There are several widely used algorithms for solving linear programs, with the most popular being the Simplex Method and interior point methods.

The Simplex Method starts at a basic feasible solution to an LP and then iteratively moves to adjacent basic feasible solutions with improved objective function value until it finds a local optimum. Since linear programs are convex optimization problems, finding a local optimum is equivalent to finding a global optimum. Recall that 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 Method is equivalent to policy iteration, but where the improvement step is only carried out for a single state. Standard policy iteration, where the improvement step is carried out for all states is equivalent to the Simplex Method with "block pivoting", meaning that instead of moving to an adjacent basic feasible solution, the algorithm may jump to a further basic feasible solution, representing a new policy with improved actions for multiple states.

Interior point methods 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. No Markov decision process algorithm is a direct analogue. Interior point methods have proven to be quite effective at solving a variety of linear programming problems, and so they may also be effective at solving certain Markov decision processes.

Software that solves a linear program will also return the optimal values of its dual variables. For example, solving the dual linear program will also give us the optimal value function. This will be useful when solving approximate dynamic programs using linear programming.

1.9 Optimality of structured policies

In this section we show how to extend results in Section **??** to infinite horizon models. We state the result in considerable generality.

1.9.1 The fundamental result

Define V^F to be a subset of V with a specific form (such as convex) and define D^F to be a subset D^{MD} in which all decision rules are structured. We say that D^F and V^F 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 a queuing service rate control model, V^F might denote the set of convex non-decreasing functions and D^F might denote monotone non-decreasing decision rules.

Theorem 1.26. Suppose:

1. V^F is non-empty,
2. D^F is non-empty and compatible with V^F,
3. $\mathbf{v} \in V^F$ implies $L\mathbf{v} \in V^F$, and
4. if $\lim_{n\to\infty} \|\mathbf{v}^n - \mathbf{v}^*\| = 0$ and $\mathbf{v}^n \in V^F$ for all $n = 0, 1, \ldots$ then $\mathbf{v}^* \in V^F$.

Then there exists an optimal stationary policy $(d^*)^\infty$ with $d^* \in D^F$.

Proof. Since V^F is non-empty, we can select a $\mathbf{v}^0 \in V^F$ and define the sequence $\mathbf{v}^{n+1} = L\mathbf{v}^n$. By induction, hypotheses 2 implies that $\mathbf{v}^n \in V^F$ for all $n = 0, 1, 2, \ldots$.

As a consequence of Theorem 1.14, $\mathbf{v}^n \to \mathbf{v}_\lambda^*$, which is the unique solution of $L\mathbf{v}_\lambda^* = \mathbf{v}_\lambda^*$. Hence by hypothesis 3, $\mathbf{v}_\lambda^* \in V^F$. Consequently there exists a $d^* \in \arg \text{c-max}_{d \in D^{\text{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}_\lambda^*\}$ that is also in D^F. □

Some comments about this theorem follow:

1. As you can see, the proof relies on the already demonstrated convergence of value iteration to the solution of the Bellman equations.

2. This result augments structural results in the finite horizon model by requiring that the limit of $(\mathbf{v}^n)$ retains the structure of all previous iterates. This usually requires results from analysis on the limiting behavior of functions.

3. Note that the result does *not* say that *all* $d \in \arg \text{c-max}_{d \in D^{\text{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}_\lambda^*\}$ are in D^F, only that there exists at least one that is in D^F.

1.9.2 An equipment maintenance example

We illustrate this result by applying it to a simple equipment maintenance model. Let $S = \{0, 1, \ldots, M\}$ represent the possible states of the equipment with higher state values representing greater deterioration. In each state $s < M$ the decision maker may either restore the equipment (action 1) to state 0 at cost K where $0 < K < M$ or let the equipment operate as is (action 0) and incur an operating cost of s units. Assume restoring the equipment takes one period and in state M, the equipment must be restored to state 0 at cost K. Hence $A_s = \{0, 1\}$ for all $s \in \{0, 1, \ldots, M-1\}$ and $A_M = \{1\}$. Assume further that

$$p(j|s,a) = \begin{cases} p & j = s+1, a = 0, s \leq M-1 \\ 1-p & j = s, a = 0, s \leq M-1 \\ 1 & j = 0, a = 1, s \leq M \\ 0 & \text{otherwise.} \end{cases}$$

Thus the equipment deteriorates one state with probability p and remains in the same state with probability $1-p$ in each period. Since the decision maker seeks to minimize the expected total discounted cost of operating the equipment over the infinite horizon we express this as a minimization problem. The Bellman equations are:

$$\begin{aligned} v(s) &= \min \{K + \lambda v(0), s + \lambda (pv(s+1) + (1-p)v(s))\}, \quad s \in \{0, 1, \ldots, M-1\} \\ v(M) &= K + \lambda v(0) \end{aligned}$$

There exists an optimal control limit policy

To apply Theorem 1.26, set V^F to be the set of non-decreasing real-valued functions on S, which is compatible with D^F equal to the set of non-decreasing Markovian deterministic decision rules on S. Note that since $A_s = \{0, 1\}$ for $s < M$, and $A_M = \{1\}$, a non-decreasing decision rule corresponds to a control limit policy.

Clearly V^F is non-empty. Suppose now that $v(s)$ is non-decreasing so that $s + \lambda(pv(s+1) + (1-p)v(s))$ is non-decreasing. Moreover, $L\mathbf{v}(s) \leq K + \lambda v(0)$ for all $s \in S$. Since the minimum of two non-decreasing functions is non-decreasing it follows from (**??**) that $L\mathbf{v}(s)$ is a non-decreasing function of s.

Finally, since the (uniform) limit of non-decreasing functions is non-decreasing, the hypotheses of Theorem 1.26 are satisfied by this example. Consequently there exists an optimal non-decreasing (control limit) policy. Note that the result also holds true if we replace s with any non-decreasing function.

We leave it as an exercise to use this result to develop an algorithm for finding an optimal control limit policy by searching within the class of control limit policies.

1.10 Application: Queuing service rate control

In this section we use the queuing service rate control model to compare solution algorithms and investigate the structure of the optimal policy.

1.10.1 The model

We consider a infinite horizon discounted version of the discrete time service rate control model described in Section **??** and analyzed in the finite horizon case in Section **??**. As before, we assume that there are three service probabilities $a_1 = 0.2$, $a_2 = 0.4$ and $a_3 = 0.6$ and that the arrival probability is $b = 0.2$. We assume further that the delay cost is $f(s) = s^2$ and the cost per period of serving at rate a_k is $m(a_k) = 5k^3$.

We truncate the state space at N and assume that the transition probabilities in state N for $k = 1, 2, 3$ are given by:

$$p(j|N, a_k) = \begin{cases} a_k & j = N-1 \\ 1 - a_k & j = N. \end{cases}$$

Since the goal is to minimize costs, we use "min" instead of the "max" in the following Bellman equation:

$$v(s) = \min_{k=1,2,3} \begin{cases} m(a_k) + \lambda((1-b)v(0) + bv(1)), & s = 0 \\ f(s) + m(a_k) + \lambda(a_k v(s-1) + (1-b-a_k)v(s) + bv(s+1)), & 0 < s < N \\ f(N) + m(a_k) + \lambda(a_k v(N-1) + (1-a_k)v(N)), & s = N \end{cases}$$

When $m(a_k)$ is increasing in k, which is reasonable for this application, there is no need to minimize in state 0 since we will always use the least expensive service rate (or turn off the server if that is possible). Hence the Bellman equation in state 0 becomes

$$v(0) = m(a_1) + \lambda((1-b)v(0) + bv(1)). \tag{1.216}$$

The beauty of this model is that as a result of the simple transition structure, we can avoid writing out the entire transition matrix when implementing an iterative algorithm.

1.10.2 Value Iteration

We explore the result of using value iteration to solve this problem. We use a span stopping criterion and estimate the value function with a lower bound approximation. The iterates of value iteration ($v^n(s)$) satisfy the easy to code recursions for $n \geq 1$:

$$v^{n+1}(0) = m(a_1) + \lambda((1-b)v^n(0) + bv^n(1))$$

$$v^{n+1}(s) = \min_{k=1,2,3}\{f(s) + m(a_k) + \lambda(a_k v^n(s-1) + (1-b-a_k)v^n(s) + bv^n(s+1))\},\ 0 < s < N$$

$$v^{n+1}(N) = \min_{k=1,2,3}\{f(N) + m(a_k) + \lambda(a_k v^n(N-1) + (1-a_k)v^n(N))\}$$

We solve the problem for $N = 50$, 200 and 1000, and $\lambda = 0.5$, 0.9 and 0.99. We initiated value iteration at $\mathbf{v}^0 = \mathbf{0}$ and chose $\epsilon = 0.0001$. We terminated value iteration when

$$\text{sp}(\mathbf{v}^{n+1} - \mathbf{v}^n) < \lambda(1-\lambda)^{-1}\epsilon$$

and used the approximation

$$\mathbf{v}_\lambda^* \approx \mathbf{v}^{n+1} + \lambda(1-\lambda)^{-1}(\underline{\mathbf{v}^{n+1} - \mathbf{v}^n})\mathbf{e},$$

which is accurate to $\pm\epsilon$ at termination.

Results

We emphasize that value iteration is only guaranteed to produce an ϵ-optimal policy. However, as our analysis of policy iteration below shows, the policy obtained by value iteration is in fact optimal.

Table 1.2 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 was negligible.

N / λ	0.5	0.9	0.99
50	27	157	384
200	31	202	754
1000	36	240	1983

Table 1.2: Iterations of value iteration required to achieve stopping criterion for discounted queuing service rate control model for different combinations of N and λ.

Figure 1.12 shows the ϵ-optimal stationary policy as a function of λ for two choices of N. Observe that in all instances the ϵ-optimal policy is **monotone** in the service rate. That is, when the queue length is longer, the server should work faster even though the cost is higher. Moreover, inspection of the policies reveals that the states at which ϵ-optimal actions change remains the same for all choices of N.

In addition observe that in each state, the ϵ-optimal service probability increases with respect to λ. For example when $s = 15$, the left graph[18] 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, 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 (checked up to 5000) is a_1.

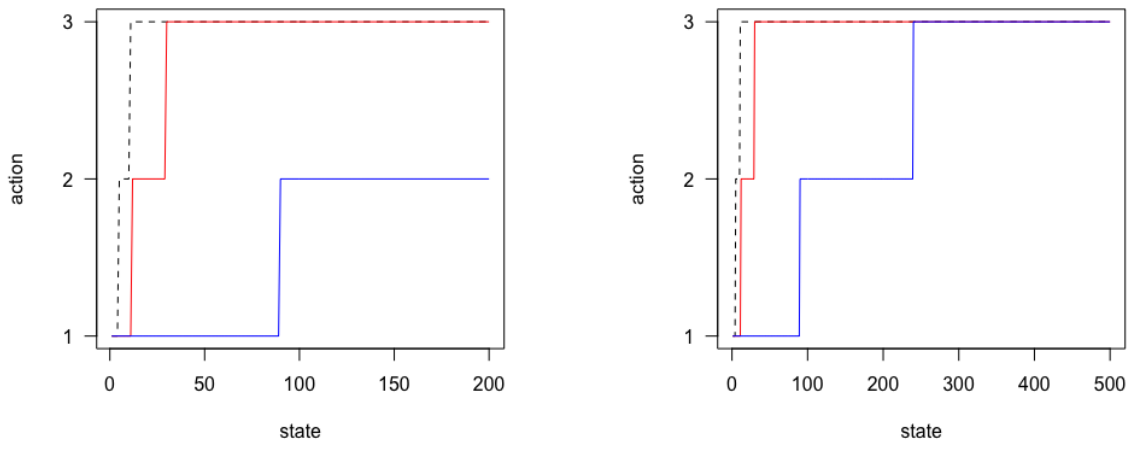


Figure 1.12: The ϵ-optimal stationary policy for the discounted queuing service rate control model as a function of λ for $\lambda = 0.5$ (red), $\lambda = 0.9$ (blue) and $\lambda = 0.99$ (dashed) truncated at $N = 200$ (left) and $N = 500$ (right).

Figure 1.13 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 our calculations using approximate dynamic programming.

1.10.3 Gauss-Seidel iteration

We solved the above instance of the discounted service rate control model using Gauss-Seidel iteration evaluating steps in the order of the states. This means we generate

[18]This observation is based on explicitly comparing the ϵ-optimal policies.

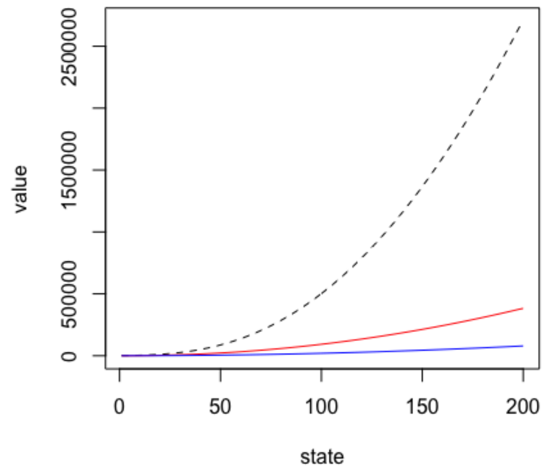


Figure 1.13: The approximation to optimal value function for the discounted queuing service rate control model as a function of λ for $\lambda = 0.5$ (red), $\lambda = 0.9$ (blue) and $\lambda = 0.99$ (dashed) truncated at $N = 200$.

updates according to:

$$v^{n+1}(0) = m(a_1) + \lambda((1-b)v^n(0) + bv^n(1)).$$

$$v^{n+1}(s) = \min_{k=1,2,3}\{f(s) + m(a_k) + \lambda(a_k v^{n+1}(s-1) + (1-b-a_k)v^n(s) + bv^n(s+1))\}, 0 < s < N$$

$$v^{n+1}(N) = \min_{k=1,2,3}\{f(N) + m(a_k) + \lambda(a_k v^{n+1}(N-1) + (1-a_k)v^n(N))\}.$$

Observe that these updates differ only from value iteration in that they use $v^{n+1}(s-1)$ instead of $v^n(s-1)$ for $s > 1$. As noted earlier, the $s = 1$ case for Gauss-Seidel and value iteration are identical.

As noted in Section 1.5.2, it is not easy to directly apply a span based stopping criterion for Gauss-Seidel iteration. Instead we use the stopping criterion

$$\|\mathbf{v}^{n+1} - \mathbf{v}^n\| < (1-\lambda)^{-1}\epsilon,$$

which by Theorem 1.15 guarantees that $\mathbf{v}^n$ is within ϵ of the optimal value function and that a stationary policy derived form a $\mathbf{v}^n$-improving decision rule is an ϵ-optimal policy.

Table 1.3 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 in spite of using the norm based stopping criterion. We observed at termination that

N / λ	0.5	0.9	0.99
50	23	90	634
200	24	90	634
1000	24	102	727

Table 1.3: 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.

$\|\mathbf{v}^{n+1} - \mathbf{v}^n\|/\|\mathbf{v}^n - \mathbf{v}^{n-1}\|$ was slightly less than λ (0.987 vs. 0.99, 0.887 vs. 0.9 and 0.443 vs. 0.5), thus justifying faster convergence.

1.10.4 Policy iteration

We now use policy iteration to find an optimal policy for the discounted queuing service rate control model with the above parameter values. We terminate iteration when the $\mathbf{v}$-improving decision rule repeats. We use the solver in R [2021] to evaluate the policy. This necessitates generating the transition probability matrix corresponding to a decision rule. Such a matrix is tri-diagonal[19] and can be generated using some programming tricks.

To explore the robustness of policy iteration we choose the arbitrary decision rule $d^0 = \{a_1, a_2, a_3, a_1, a_2, a_3, \dots\}$ as the first to evaluate. As above we truncated the state space at $N = 50, 200, 1000$ and chose $\lambda = 0.5, 0.9, 0.99$. Results including the number of iterations required and the optimal policy are reported in Table 1.4.

Table 1.4 shows that in all instances except the first, three iterations (policy evaluations and minimizations) were required to find an optimal policy. Since the optimal policy for this model was monotone increasing in the service rate as a function of the queue length, we represented the optimal policies by the change points between actions as described 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 we solved the problem with 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 $\lambda = 0.99$. We noted that when choosing the "argmax" in the improvement step using

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

the quantities in the brackets in state 4 were 2080.04 when $a = a_1$ and 2080.87 when $a = a_2$ so that a_1 was selected when $N = 1000$.

[19]A matrix is *tri-diagonal* if all non-zero entries lie on the sub-diagonal and super-diagonal. The *row*$(\cdot)$ and *column*$(\cdot)$ functions in R [2021] 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
1000	0.50	3	89	239
1000	0.90	3	11	29
1000	0.99	3	3*	10

Table 1.4: 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. We refer to the starred cell in text, "n/a" indicates that this switch does not occur before the truncation point N.

To explore the performance of policy iteration, we solved a larger problem instance with more states and actions. In it, $A_s = \{a_1, \ldots, a_6\}$, corresponding to service rates $\{0.2, 0.3, \ldots, 0.7\}$. The cost of using rate a_k was $m(a_k) = 2k^3$, $\lambda = 0.4$ and $N = 5000$.

Policy iteration found an optimal policy in 2 iterations, starting from the arbitrary decision rule $d^0 = (a_1, a_6, a_1, a_6, \ldots)$. As hypothesized the execution time was not insignificant because of the need to solve a 5000×5000 linear system of equations. As shown in Figure 1.14 the optimal policy was monotone and the value function non-decreasing in the queue length.

1.10.5 Modified policy iteration

We solved several versions this model using MPI with our main focus being comparison to the computational effort of value iteration and the effect of the truncation sequence $\{m_n\}$ on computational effort. We used the result in Theorem 1.13 to obtain an ϵ-optimal policy at termination.

In our implementation we found it convenient to start with a decision rule d' and implemented the evaluation step in matrix form by applying the recursion

$$\mathbf{v} \leftarrow \mathbf{r}_{d'} + \lambda \mathbf{P}_{d'} \mathbf{v} \tag{1.217}$$

$m_n + 1$ times. We implemented the improve step in component form and noted that computing

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

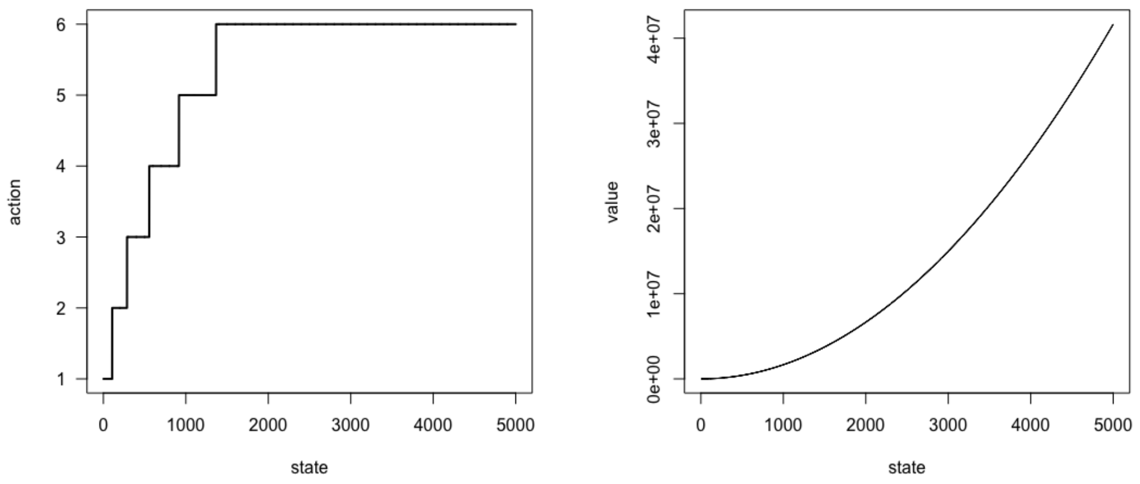


Figure 1.14: Optimal policy (left) and value function (right) obtained when solving the larger instance of the service rate control problem.

for all $s \in S$ yielded **both** an updated decision rule d' and $L_{d'}\mathbf{v}$, Hence at successive iterations we needed to apply (1.217) m_n starting with $L_{d'}\mathbf{v}$ obtained from the improvement step.

In describing results, we note that one improvement step required the same effort as

$$\frac{\sum_{s\in S}|A_s|}{|S|}$$

evaluation steps[20]. In this example, $|A_s| = 3$ for all $s \in S$, which means that the improvement step requires three times more effort than one evaluation update using (1.217). If in this example the algorithm terminated after M iterations, the total effort would be equivalent to $\sum_{n=1}^{M} m_n + 3M$ iterations of (1.217), which when $m_n = 5$ and $N = 30$ would be 240. The following table summarizes the effort as a function of the truncation sequence for the instance when the state space is truncated at $N = 200$ and $\lambda = 0.9$. We initialized MPI with the arbitrary decision rule $d = (a_1, a_2, s_3, a_1, a_2, a_3, \dots)$.

Table 1.5 shows that all implementations of modified policy iteration required less effort than value iteration ($m_n = 0$) for solving the discounted queuing control model. Note than the number of iterates for value iteration differed from that in Table 1.2 because the implementation here 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

[20]We could define the computation in (1.217) as one unit of effort. On this basis the improve step would require three units of effort when there are three possible service rates. Thus MPI trades off the number of "cheap" evaluation steps with "more expensive" improvement steps

Truncation sequence (m_n)	Number of iterations, M	"Effort"
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 1.5: 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 varied m_n with n.

sequences n and $\text{int}(n^{0.5})$[21] required more effort than the decreasing truncation sequence $\max(30 - n, 0)$.

We then solved the $N = 1000$ state version with $\lambda = 0.9$ using results above as guidance. We found that the effort was lowest when using the *decreasing* sequence $m_n = \max(30 - n, 0)$ which required $M = 10$ iterations and an effort of 255. In contrast the effort for $m_n = 20$ was 299 and that of value iteration was 783.

We then solved a $5000, 10000$ and 15000 state model with 6 actions per state as analyzed in the policy iteration section. We initialized the calculations with the decision rule $d = (a_1, a_2, s_3, a_1, a_2, a_3, \dots)$ and set $= 0.00001$. Value iteration required 293 iterations or an equivalent effort of 1758. Moreover it required 21.69 minutes, to obtain an ϵ-optimal solution. In contrast, modified policy iteration with $m_n = \max(30 - n, 0)$ required 12 iterations or equivalently an 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 1.6 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 3 iterations to find an optimal policy. The majority of the execution time was spent in the evaluation step solving

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

[21] $\text{int}(x)$ denotes the integer part of x.

N	Policy iteration	Policy iteration	Modified policy iteration
	total time	evaluation time	total time
5000	2.40	2.00	1.38
10000	18.92	17.34	16.17
15000	71.76	67.03	28.23

Table 1.6: Comparison of computation times (in minutes) for policy iteration and modified policy iteration with decreasing truncation series as a function of problem size. Total time solving systems of linear equations in the evaluation step of policy iteration appears in column 3.

1.10.6 Linear Programming

We now formulate and solve the queuing service rate control model as a linear program. We begin with primal formulation (1.164). Let $\alpha(s) > 0, s \in \{0, 1, \ldots, N\}$ satisfy $\sum_{s=0}^{N} \alpha(s) = 1$. Since this is a cost minimization problem, it is natural to formulate the primal as the maximization problem[22]. Set $c(s, a_k) = f(s) + m(a_k) = s^2 + 5k^3$.

The model follows:

$$\begin{aligned}
\text{maximize} \quad & \alpha(0)v(0) + \ldots + \alpha(N)v(N) \\
\text{subject to} \quad & v(0) - \lambda((1-b)v(0) + bv(1)) \leq c(0, a_k), \quad k = 1, 2, 3, \\
& v(s) - \lambda(a_k v(s-1) + (1 - a_k - b)v(s) + bv(s+1)) \leq c(s, a_k), \quad k = 1, 2, 3, \; s = 1, \ldots, N - \\
& v(N) - \lambda(a_k v(N-1) + (1 - a_k)v(N)) \leq c(s, a_k), \quad k = 1, 2, 3.
\end{aligned}$$

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 we need only retain the tightest constraint corresponding to the least costly action, a_1 (see equation (1.216)). Moreover with a little algebra the constraints can be simplified considerably. For illustrative purposes, we include all constraints as is.

As noted earlier, we prefer to solve the problem through its dual (1.8.2):

[22]Alternatively, we could formulate it as a minimization problem, but this would require setting rewards equal to negative costs.

$$\begin{aligned}
\text{minimize} \quad & \sum_{s=0}^{N}\sum_{k=1}^{3} c(s,a_k)x(s,a_k) \\
\text{subject to} \quad & \sum_{k=1}^{3} x(0,a_k) - \lambda\left(\sum_{k=1}^{3}((1-b)x(0,a_k)+a_kx(1,a_k))\right) = \alpha(0), \\
& \sum_{k=1}^{3} x(s,a_k) - \lambda\left(\sum_{k=1}^{3}(bx(s-1,a_k)+(1-a_k-b)x(s,a_k)+a_kx(s+1,a_k))\right) = \alpha(s), \quad s = \\
& \sum_{k=1}^{3} x(N,a_k) - \lambda\left(\sum_{k=1}^{3}(bx(N-1,a_k)+(1-a_k)x(N,a_k))\right) = \alpha(N) \\
& x(s,a_k) \geq 0, \quad k=1,\dots,3,\ s=0,\dots,N,
\end{aligned}$$

Observe that the dual has $3(N+1)$ variables, $N+1$ equalities and $3(N+1)$ non-negativity constraints. *When coding, we found it easier to write out the primal and then form the dual by transposition.*

We solve the model for $N = 50$ and $\lambda = 0.9$ and $\alpha(s) = 1/(N+1)$. To obtain an optimal policy directly, we solve the dual. Since there are $N+1$ equality constraints, an optimal basic feasible solution has at most $N+1$ positive variables and at least $2(N+1)$ variables equal to zero. We obtain a solution with exactly $N+1$ positive variables:

$$x(s,a_k) > 0 \quad \text{when} \begin{cases} k=1 \text{ and } s=0,\dots,10 \\ k=2 \text{ and } s=11,\dots,28 \\ k=3 \text{ and } s=29,\dots,50. \end{cases}$$

The decision rule corresponding to an optimal policy sets $d^*(s) = a_k$ when $x(s,a_k) > 0$. The optimal policy $(d^*)^\infty$ defined in this way agrees with the optimal policy found using policy iteration when $\lambda = 0.9$ and $N = 50$. Further we observed that:

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,
3. the optimal policy and primal value functions did not vary with $\alpha(s)$, and
4. we were able to solve considerably larger instance of the problem easily.

1.10.7 Concluding remarks on queuing control application

In this section we solved the queuing service rate control model using the algorithms described in the body of the chapter. We observed that:

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 outperformed fixed and increasing truncation sequences.

5. Modified policy iteration (with truncation sequence $m_n = \max(30 - n, 0)$) was the fastest method for finding optimal policies in larger instances ($N \geq 5000$).

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.

1.11 Application: Clinical decision making

We provide a numerical example of a discounted version of the liver transplantation application in Section **??**. Our main purposes in providing this example are:

- to analyze a model with a more complex transition structure,
- to illustrate how an insightful graphical representation of optimal policies in the presence of a multidimensional state space,
- to provide an optimal policy with a clinically meaningful structure and
- to delve deeper into an innovative application,

In this application, discounting is appropriate since we either view discounting as modeling patient death due to causes independent of death from organ failure or from the perspective that a day alive today is worth more than a potential day alive in the future.

1.11.1 The Bellman equation

Given the model components defined previously, the Bellman equation can be written as

$$v(h,l) = \max\{R(h,l), \sum_{\substack{h' \in S_H \\ l' \in S_L}} (1+\lambda v(h',l))q_h(h'|h)w(l'|h) \\ + (1+\lambda v(h',\Phi))q(h'|h)\phi(h) + \lambda v(\Delta)\delta(h)\} \tag{1.218}$$

for $h \in S_H, l \in S_L$,

$$v(h,\Phi) = \sum_{\substack{h' \in S_H \\ l' \in S_L}} (1+\lambda v(h',l'))q_h(h'|h)w(l'|h) + (1+\lambda v(h',\Phi))q(h'|h)\phi(h) + \lambda v(\Delta)\delta(h)\} \tag{1.219}$$

and $v(\Delta) = 0$[23].

In the maximization, the first term corresponds to accepting the offered organ (a_t) while the second term corresponds to the decision to not accept an organ and wait one period (a_w). 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[24].

1.11.2 A numerical example

Without access to primary data used in Alagoz et al. [2007] we chose parameters that are somewhat realistic but most importantly produce an optimal policy with a similar structure to that in the article.

Set $H = 20$ and $L = 10$. Let $\lambda = 0.99978$ be the daily discount rate corresponding 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. We set

$$R(h,l) = \alpha[365(14 - .25(h-1) - .6(l-1))].$$

[23]Because Δ is a zero-reward absorbing state, this application could be modelled without discounting as a transient model as described in Chapter **??**

[24]This observation simplifies coding the algorithm. Note we observed a similar structure in the online dating model of Section **??**.

With $\alpha = 0.15$, mean survival varies between 7 and 25.6 months[25]

We propose transition probabilities without transplant to be

$$\delta(20) = .001 \text{ and } q(20|20) = .999$$

and for $h = 1, \dots, 19$;

$$\delta(h) = .0003h \quad \text{and} \quad q(h'|h) = \begin{cases} 0.997 - 0.0004h & h' = h \\ 0.003 + 0.0001h & h' = h + 1 \end{cases}$$

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.[26]. Note that using properties of transient Markov chains (See Appendix **??**) we can compute the expected number of days to death without transplant directly from solving $(\mathbf{I} - \mathbf{Q})\mathbf{u} = \mathbf{1}$ where $\mathbf{Q}$ denotes the matrix with components $q(h'|h)$. We find that expected survival times without transplant are between 3.3 months in health state 20 and 35.2 months in health state 1[27]

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, We choose the probability of a patient in health state not being offered an organ to be

$$\phi(h) = 1 - 0.01h \quad \text{for} \quad h = 1, \dots, 20$$

and

$$w(l|h) = 0.001h \quad \text{for} \quad h = 1, \dots, 20,\ l = 1, \dots, 10.$$

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.

1.11.3 Numerical solution

We solve this model using policy iteration with iterative policy evaluation[28] and begin the iteration with the decision rule $d(s) = a_w$, that is, reject an offered organ in every state (h, l) for $h \in S_H$ and $L \in S_L$.

[25]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.

[26]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*

[27]Literature suggests that 3-month survival rates vary between 27% and 98% depending on patient health state (MELD score).

[28]This is equivalent to modified policy iteration with a very high order (20000). We use iterative evaluation for coding simplicity since it is tedious to manipulate matrices with state vectors in R [2021]

Note that under the policy that uses d, $v(h, l)$ does not depend on l so that we write it as $u(h)$. We find $u(h)$ by solving

$$(\mathbf{I} - \lambda \mathbf{Q})\mathbf{u} = \mathbf{1} \tag{1.220}$$

where $\mathbf{I}$ denotes an $|S_H| \times |S_H|$ identity matrix, $\mathbf{Q}$ denotes an $|S_H| \times |S_H|$ matrix with elements $q(h'|h)$[29] and $\mathbf{u}$ denotes an $|S_H|$-component vector with entries $u(h)$.

We represent $u(h)$ by the solid line in Figure 1.15. We 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[30].

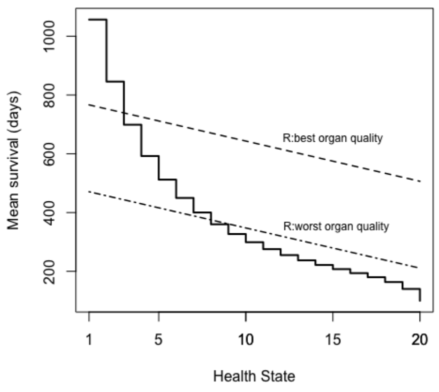


Figure 1.15: Comparison of mean (discounted) survival days $u(h)$ without transplant (solid line) and specified survival post transplant $R(h, l)$ for two organ qualities (dashed lines).

To implement the first improvement step we set $v(h, l) = u(h)$ for $L = 1, \ldots, L$ and we use the right hand side of (1.218) to identify an improvement. We obtain convergence after 2 iterations.

1.11.4 Interpretation of results

We discuss properties of the optimal policy which is displayed graphically in Figure 1.16a. Observe that regardless of organ type, it is optimal to wait in health states

[29]This matrix corresponds to transient states of the health state change process.

[30]We emphasize that our choice of $R(h, l)$ was not based on data but specified to generate the pattern in Figure 1.15.

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 acceptance region decreases 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 acceptance region decreases with organ quality.

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. Using methods discussed in Section 1.9, Alagoz et al. [2007] establish this result analytically under assumptions on the model components.

Figure 1.16b shows discounted survival times for the no transplantation option (as in Figure 1.15 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 such a patient will get improved survival because when he/she reaches a poorer health state, the improved survival under transplantation $R(h, l)$ will replace the survival under no transplantation $u(h)$.

1.11.5 Discussion

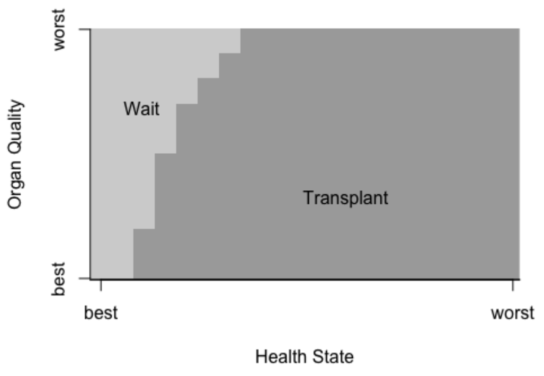
This section has formulated and used Markov decision processes to analyze a stylized version of a realistic clinical decision problem. It provides a intuitive graphical representation of an optimal policy and interprets it from a clinical perspective. Moreover by comparing and interpreting survival (value functions) under several options, it shows the benefits of using a dynamic decision making as opposed to a myopic approach.

1.12 Technical Appendices to Chapter 1

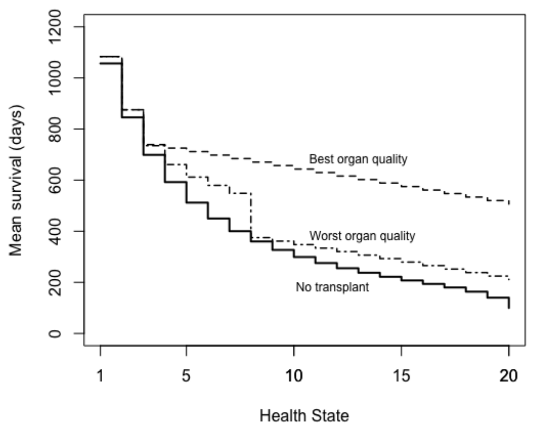
1.12.1 Proof of Theorem 1.1

We will prove this result in the more general case where the reward depends on the subsequent state, that is, $v_\lambda^\pi(s)$ may be written as

$$v_\lambda^\pi(s) = \sum_{n=1}^{\infty} \sum_{j \in S} \sum_{a \in A_j} \lambda^{n-1} r(j, a, k) p^\pi(X_n = j, Y_n = a, X_{n+1} = k | X_1 = s). \tag{1.221}$$



(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 1.16: Graphical representation of the optimal policy and value function in the liver transplantation application.

Theorem 1.1 follows immediately from the following lemma[31]. Moreover this lemma will provide the basis for establishing an analogous result to that in Theorem 1.1 for the expected total reward and average reward criteria.

Lemma 1.9. Let $\pi = (d_1, d_2, \ldots) \in \Pi^{\text{HR}}$. For each $s \in S$ and $n = 1, 2, \ldots$, there exists $\pi' = (d'_1, d'_2, \ldots) \in \Pi^{\text{MR}}$ such that

$$p^{\pi}(X_n = j, Y_n = a, X_{n+1} = k | X_1 = s) = p^{\pi'}(X_n = j, Y_n = a, X_{n+1} = k | X_1 = s). \tag{1.222}$$

Proof. Fix $s \in S$. Let $\pi' = (d'_1, d'_2, \ldots) \in \Pi^{\text{MR}}$ be such that decision rule d'_n is defined by

$$w_{d'_n}(j, a) := p^{\pi}(Y_n = a | X_n = j, X_1 = s), \tag{1.223}$$

for $j \in S, a \in A_j, k \in S$ and $n = 1, 2, \ldots$. In other words, this Markovian randomized decision rule induces a probability distribution over the actions that matches the probability distribution induced by the given history-dependent randomized decision rule d_n.

Now, we prove the stated result by induction on n. Clearly, the result is true for $n = 1$. Let $n > 1$ and suppose (1.222) holds for $t = 1, \ldots, n$. Using properties of

[31]This result was proved by Strauch [1966] in greater generality.

conditional expectations

$$\begin{aligned}&p^\pi(X_{n+1}=k, X_n=j, Y_n=a|X_1=s)= \\ &\quad p^\pi(X_{n+1}=k|X_n=j,Y_n=a,X_1=s)p^\pi(Y_n=a,|X_n=j,X_1=s)p^\pi(X_n=j|X_1=s).\end{aligned} \tag{1.224}$$

We show that each expression in equation (1.224) with respect to $\pi \in^{\text{HR}}$ is equal to the corresponding expression for $\pi' \in^{\text{MR}}$ as follows.

Note that the first term,

$$p^\pi(X_{n+1}=k|X_n=j,Y_n=a,X_1=s)=p(k|j,a)=p^{\pi'}(X_{n+1}=k|X_n=j,Y_n=a,X_1=s),$$

since conditional on $X_n=j$ and $Y_n=a$, X_{n+1} is independent of X_1.

Next consider the second term. Since π' is Markovian, it follows from (1.223) that

$$p^{\pi'}(Y_n=a|X_n=j,X_1=s)=p^{\pi'}(Y_n=a|X_n=j)=w_{d'_n}(j,a)=p^\pi(Y_n=a|X_n=j,X_1=s) \tag{1.225}$$

for all n. Thus, what remains is to show

$$p^\pi(X_n=j|X_1=s)=p^{\pi'}(X_n=j|X_1=s). \tag{1.226}$$

This follows from noting that

$$\begin{aligned}p^\pi(X_n=j|X_1=s)&=\sum_{l\in S}\sum_{a\in A_l}p(j|l,a)p^\pi(X_{n-1}=l,Y_{n-1}=a|X_1=s) &(1.227)\\ &=\sum_{l\in S}\sum_{a\in A_l}p(j|l,a)p^{\pi'}(X_{n-1}=l,Y_{n-1}=a|X_1=s) &(1.228)\\ &=p^{\pi'}(X_n=j|X_1=s), &(1.229)\end{aligned}$$

where the second equality is a consequence of the induction hypothesis. □

We emphasize the following subtle point regarding interpreting the result of the lemma. 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 the following example shows, it does not guarantee that same policy will achieve this result for all states.

Example 1.18. To show that the corresponding Markovian policy depends on the initial system state, consider a deterministic model with two states, u and w and two actions a and 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.

Puterman [1994] provides an example illustrating this construction in a stochastic problem with history dependent randomized policies on pp 134-135.

1.12.2 Bounds for Gauss-Seidel iteration

In this section, we generalize the arguments in Section 1.4.2 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 $k = 1, \ldots, M$

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

We now decompose $\mathbf{P}_{d'}$ in to its lower and upper triangular parts, $\mathbf{P}_{d'} = \mathbf{P}^L_{d'} + \mathbf{P}^U_{d'}$, where

$$\mathbf{P}^L_{d'} = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ p_{21} & 0 & 0 & \ldots & 0 & 0 \\ p_{31} & p_{32} & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{M-1,1} & p_{M-1,2} & p_{M-1,3} & \cdots & \ddots & 0 \\ p_{M1} & p_{M2} & p_{M3} & \cdots & p_{M,M-1} & 0 \end{bmatrix}$$

and

$$\mathbf{P}^U_{d'} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1,M-1} & p_{1M} \\ 0 & p_{22} & p_{23} & \cdots & p_{2,M-1} & p_{2M} \\ 0 & 0 & p_{33} & \cdots & p_{3,M-1} & p_{3M} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \ddots & p_{M-1,M} \\ 0 & 0 & 0 & \ldots & 0 & p_{MM} \end{bmatrix}.$$

Then in vector form

$$G\mathbf{v} = \underset{d\in D^{\text{MD}}}{\text{c-max}} \left\{ \mathbf{r}_d + \lambda \mathbf{P}^L_d G\mathbf{v} + \lambda \mathbf{P}^U_d \mathbf{v} \right\} = \mathbf{r}_{d'} + \lambda \mathbf{P}^L_{d'} G\mathbf{v} + \lambda \mathbf{P}^U_{d'} \mathbf{v}.$$

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, we obtain

$$(\mathbf{I} - \lambda \mathbf{P}_{d'}^L) G\mathbf{v} = \mathbf{r}_{d'} + \lambda \mathbf{P}_{d'}^U \mathbf{v}.$$

Since $(\mathbf{I} - \lambda \mathbf{P}_{d'}^L)^{-1}$ exists, we have the following elegant representation for G

$$\begin{aligned} G\mathbf{v} &= (\mathbf{I} - \lambda \mathbf{P}_{d'}^L)^{-1} \mathbf{r}_{d'} + (\mathbf{I} - \lambda \mathbf{P}_{d'}^L)^{-1} \lambda \mathbf{P}_{d'}^U \mathbf{v} \\ &= \underset{d \in D^{\mathrm{MD}}}{\text{c-max}} \left\{ (\mathbf{I} - \lambda \mathbf{P}_d^L)^{-1} \mathbf{r}_d + (\mathbf{I} - \lambda \mathbf{P}_d^L)^{-1} \lambda \mathbf{P}_d^U \mathbf{v} \right\}. \end{aligned} \tag{1.230}$$

We introduce the following further notation for consistency with literature, to simplify exposition and allow for some generalizations. For $d \in D^{\mathrm{MD}}$, we define two matrices[32] as follows:

$$\mathbf{Q}_d := \mathbf{I} - \lambda \mathbf{P}_d^L \quad \text{and} \quad \mathbf{R}_d := \lambda \mathbf{P}_d^U.$$

Hence we can express G in this notation as

$$G\mathbf{v} = \underset{d \in D^{\mathrm{MD}}}{\text{c-max}} \{ \mathbf{Q}_d^{-1} \mathbf{r}_d + \mathbf{Q}_d^{-1} \mathbf{R}_d \mathbf{v} \}. \tag{1.231}$$

We now generalize Lemma 1.5 to the operator G. The proof is identical to that above but replaces $L\mathbf{v}$ by the representation (1.231) for $G\mathbf{v}$.

Lemma 1.10. Let $\mathbf{u} \in V$ and $\mathbf{u} \in V$. If $d_{\mathbf{v}}$ attains the maximum of $G\mathbf{v}$ in (1.231), then

$$G\mathbf{v} - \mathbf{v} \geq G\mathbf{u} - \mathbf{u} + (\mathbf{Q}_{d_\mathbf{v}}^{-1} \mathbf{R}_{d_\mathbf{v}} - \mathbf{I})(\mathbf{u} - \mathbf{v}). \tag{1.232}$$

We apply it in the same way as in the derivation of Theorem 1.12 to obtain the following bounds. The quantity $(\mathbf{I} - \mathbf{Q}_d^{-1} \mathbf{R}_d)^{-1}$ replaces $(\mathbf{I} - \lambda \mathbf{P}_d)^{-1}$ in the derivation and Lemma 1.2 provides its representation as

$$(\mathbf{I} - \mathbf{Q}_d^{-1} \mathbf{R}_d)^{-1} = \sum_{n=0}^{\infty} (\mathbf{Q}_d^{-1} \mathbf{R}_d)^n. \tag{1.233}$$

Lemma 1.11. For any $d \in D^{\mathrm{MR}}$, $\|\mathbf{Q}_d^{-1} \mathbf{R}_d\| = \lambda$.

[32]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.

Proof. Since $\mathbf{Q}_d^{-1} = (\mathbf{I} - \lambda \mathbf{P}_d^L)^{-1} = \sum_{n=0}^{\infty} (\lambda \mathbf{P}_d^L)^n \geq \mathbf{I}$,

$$(\mathbf{I} - \lambda \mathbf{P}_d) \leq \mathbf{Q}_d^{-1}(\mathbf{I} - \lambda \mathbf{P}_d) = \mathbf{Q}_d^{-1}(\mathbf{Q}_d - \mathbf{R}_d) = \mathbf{I} - \mathbf{Q}_d^{-1}\mathbf{R}_d.$$

Hence $\lambda \mathbf{P}_d \geq \mathbf{Q}_d^{-1}\mathbf{R}_d \geq \mathbf{0}$. Therefore $\lambda = \|\lambda \mathbf{P}_d\| \geq \|\mathbf{Q}_d^{-1}\mathbf{R}_d\|$. 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 result follows. □

We will use this lemma in the proof of the following theorem but it also guarantees the row sums of $\mathbf{Q}_d^{-1}\mathbf{R}_d$ are all less than or equal 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 probability matrix $\mathbf{P}$, let $\mu(\mathbf{P})$ denote the minimum row sum of $\mathbf{P}$. Of course $\|\mathbf{P}\|$ equals its maximum row sum. Note that the proof differs slightly from that of Theorem 1.12 because the row sums of $\mathbf{Q}_d^{-1}\mathbf{R}_d$ might not all equal λ.

Theorem 1.27. 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 $\mathbf{v} \in V$,

$$\mathbf{v} + (1-\lambda)^{-1}(\overline{G\mathbf{v} - \mathbf{v}})\mathbf{e} \geq G\mathbf{v} + \lambda(1-\lambda)^{-1}(\overline{G\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v}_\lambda^* \geq \mathbf{v}_\lambda^{d_\mathbf{v}^\infty} \tag{1.234}$$

$$\geq G\mathbf{v} + \alpha(1-\alpha)^{-1}(\underline{G\mathbf{v} - \mathbf{v}})\mathbf{e} \geq \mathbf{v} + (1-\alpha)^{-1}(\underline{G\mathbf{v} - \mathbf{v}})\mathbf{e}$$

Proof. Following the argument used to obtain the upper bound in the proof of Theorem 1.12 and (1.233), 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 1.11, $\|\mathbf{Q}_{d_\mathbf{v}}^{-1}\mathbf{R}_{d_\mathbf{v}}\| = \lambda$ 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 1.12.

To obtain the lower bound, apply Lemma 1.10 and (1.233) 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 1.12 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. We illustrate the behavior of these bounds in a case where we can evaluate α a priori.

Example 1.19. We return to the two-state example. Since there are only four Markovian decision rules, we can evaluate α when $\lambda = 0.9$. 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 1.11. Also we see that $\alpha = 0.54$ in this example. Since α corresponds to the optimal policy, we would expect the lower bound to be applicable.

We now show how these bounds behave empirically in the following table of iterates of Gauss-Seidel with $\mathbf{v}^0 = \mathbf{0}$.

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 we replaced λ by α in the upper bound and α by λ in the lower bound, neither were valid.

1.12.3 Convergence of policy iteration in non-finite models*

Figure 1.17 shows the logic underlying the Newton's method representation for policy iteration, illustrates why it converges in a few iterations in a finite model and provides the basis for proofs of convergence of modified policy iteration, to be discussed in the next section.

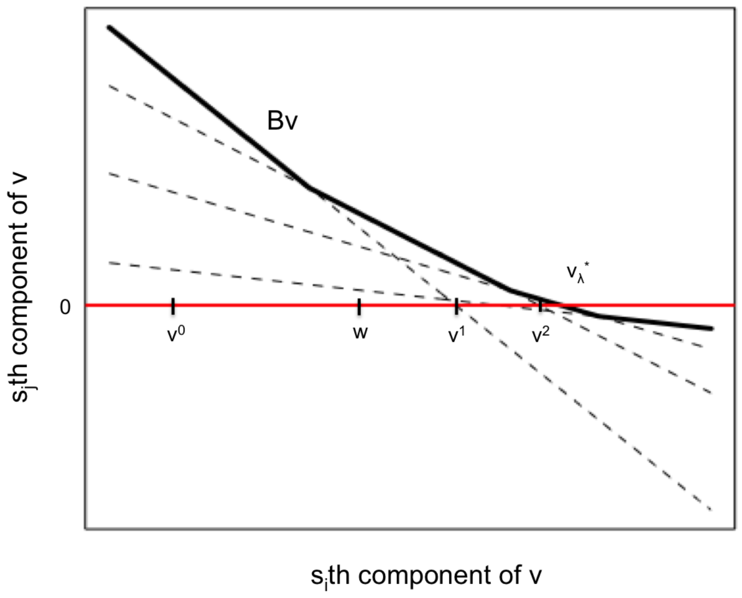


Figure 1.17: The bold line represents $B\mathbf{v}(s_j)$ as a function of $\mathbf{v}(s_i)$ for two states s_i and s_j. The dashed lines represent $B_d\mathbf{v}(s_j)$ as a function of $B\mathbf{v}(s_i)$ for each of four decision rules. When policy iteration starts at $\mathbf{v}^0$, the improvement step identifies a v^0-improving decision rule with support $B_{d_{v^0}}\mathbf{v}$. In the evaluation step, it solves $B_{d_{v^0}}\mathbf{v} = \mathbf{0}$ 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 we were able to identify $\mathbf{w}$ with less effort than solving $B_{d_{v^0}}\mathbf{v} = \mathbf{0}$, the w-improving decision rule would be identical to that at $\mathbf{v}^1$. The modified policy iteration method seeks to do this.

Iteratively applying Lemma 1.6 and using the established convergence of value iteration yields the following important convergence result for policy iteration. (See the proof of Theorem 1.17 for a detailed proof using the same logic.)

Theorem 1.28. Suppose $B\mathbf{v}^0 \geq \mathbf{0}$ and for all $\mathbf{v} \in V$ there exists a

$$d_{\mathbf{v}} \in \arg\max_{d \in D^{\mathrm{MD}}} \{\mathbf{r}_d + \lambda \mathbf{P}_d \mathbf{v}\}. \tag{1.235}$$

Then the sequence of iterates $(\mathbf{v}^n)_{n=0}^{\infty}$ 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 you can start policy iteration in step 3 with any $\mathbf{v}^0$ satisfying $B\mathbf{v}^0 \geq \mathbf{0}$. Recall that if $B\mathbf{v}^0 \geq \mathbf{0}$, $\mathbf{v}^0 \leq \mathbf{v}_\lambda^*$.

The crucial hypothesis is that the maximum is attained in (1.235). 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.

Note that in non-finite circumstances, PIA may never satisfy the stopping criterion $d'' = d'$. In such cases you can terminate the algorithm using a stopping criterion based on $\|\mathbf{v}^{n+1} - \mathbf{v}^n\|$ or $\mathrm{sp}(\mathbf{v}^{n+1} - \mathbf{v}^n)$. The value of the decision rule at termination provides the tightest lower bound while the tightest upper bound from Theorem 1.12 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.

As we noted above, 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 1.29. Suppose for all $\mathbf{u}$ and $\mathbf{v}$ in V, there exists a $K > 0$ for which

$$\|\mathbf{P}_{d_{\mathbf{u}}} - \mathbf{P}_{d_{\mathbf{v}}}\| \leq K\|\mathbf{u} - \mathbf{v}\| \tag{1.236}$$

then

$$\|\mathbf{v}^{n+1} - \mathbf{v}_\lambda^*\| \leq \frac{K\lambda}{1-\lambda}\|\mathbf{v}^n - \mathbf{v}_\lambda^*\|^2. \tag{1.237}$$

We refer the reader to Section 6.4.4 in Puterman [1994] for a proof of this Theorem and an example illustrating quadratic convergence and conditions on the model components that ensure (1.236) holds.

Note that condition (1.236) does not hold in finite state and action models because $\|\mathbf{P}_{d_{\mathbf{u}}} - \mathbf{P}_{d_{\mathbf{w}}}\|$ will remain constant when $\mathbf{P}_{d_{\mathbf{u}}} \neq \mathbf{P}_{d_{\mathbf{w}}}$ for $\|\mathbf{u} - \mathbf{w}\|$ small. (Refer to $\mathbf{u}$ and $\mathbf{w}$ in Figure 1.8 on either side of the value where components of $\mathbf{r}_d + (\lambda \mathbf{P}_d - \mathbf{I})\mathbf{v}(s)$ intersect for two decision rules.)

Sufficient conditions for (1.236) to hold[33] 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 $a \in A_s$.

1.13 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 Blackwell [1965] were seminal papers on the theory of discounted Markov decision process models, with the former being the most accessible 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. Our discussion of strict improvement for policy iterations is motivated by results on pg. 40-41 in Derman [1970].

The discussion of the span follows Section 6.6.1 of Puterman [1994]; its use in Markov decision processes originates with Bather [1973]. 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. Our proof that the Gauss-Seidel operator G is a contraction uses concepts in Bertseksas [2012]. The bounds for GS in Section 1.12.2 are new and use several results from Puterman [1994] regarding representing Gauss-Seidel updates as a regular splitting.

[33]Results that describe the form of v-improving decision rules are referred to as *selection theorems*.

The concepts of hybrid algorithms and details on asynchronous value iteration may be found in Bertseksas [2012].

Modified policy iteration was independently proposed by Puterman and Shin [1978] and van Nunen [1976]. Our development of modified policy iteration 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 Bertseksas [2012], where he refers to it as *optimistic* policy iteration.

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].

1.14 Exercises

1. We revisit Exercise **??** from Chapter 2. Assum e $\lambda = 0.9$.
   (a) Write out the optimality equation.
   (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 **??** 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) Stop evaluation with a span based criterion with δ_n increasing and decreasing.

(e) Integrate the algorithm with Gauss-Seidel updating where appropriate.

(f) 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 **??** with a finite warehouse of capacity 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 you results compare to those when the stock arrives instantaneously?

4. Consider the replacement model in Section 1.9.2 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 **??** in Chapter 3) using $\lambda = 0.9$.

6. Consider a discounted infinite-horizon version of the queuing admission control model of Section **??**. Assume the arrival probability is $q = 0.3$, 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 equations 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 equations 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 or 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)[34] 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. We seek to evaluate the policy that stops in state s_1 only. 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 $\lambda = 0.999$.

[34]This example is a good jumping off point for analyzing the transplant model in the next few exercises.

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. Evaluating this policy using the component form recursion

$$v'(s) = r(s, d(s)) + \lambda \sum_{s' \in S} p(j|s, d(s))v(s')$$

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 1.11 with $H = 2$ and $L = 2$.[35]

(a) Depict the model symbolically in the form of Figure **??** 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 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 you choice and the discount rate $\lambda = 0.9997$, evaluate the policy that accepts an organ only in state $(2, 2)$ in three ways:

i. By solving $(\mathbf{I} - \lambda \mathbf{P})\mathbf{v} = \mathbf{r}$.
ii. Using an iterative scheme in vector form $\mathbf{v}' = \mathbf{r} + \lambda \mathbf{P}\mathbf{v}$.
iii. Using a similar recursion in component form in which the summations are computed recursively.
iv. How many iterations are required to accurately compute the exact value found by solving the linear equation.

11. We revisit the liver transplantation application formulated and analyzed in Section 1.11 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 1.16a. 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 the specification in Section 1.11 when only a quality 1 organ is available.

[35] We used this example to debug our code for solving the model in Section 1.11.

(b) Repeat your analysis when only a quality 10 organ is available and compare your results.

(c) Provide conditions under which the optimal policy is a control limit in the health state. Prove that the optimal policy has this form.

(d) Solve a varaint of the problem the model when there is a single health state (state 6) and multiple organ types. Be sure to redefine transition probabilities.

13. (A really neat problem)[36] 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 (1.14) in the sum over n is equal to (1.17).

15. Prove that $\text{sp}(\mathbf{v})$ is a semi-norm and in addition satisfies properties (1)-(3) following its definition.

16. Evaluate γ and γ' for each of the three other Markovian deterministic policies using the approach in Example 1.7 and equations (1.104) and (1.105). Also, compute the eigenvalues of each of the transition matrices and compare them to γ.

17. Evaluate the bounds in Example 1.9 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.

18. Implement value iteration in Example 1.11 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.

19. 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.

20. Prove a variant of Proposition 1.7 when there are strict improvements in two states.

21. Provide a formal proof of Theorem 1.28 using induction and Lemma 1.6.

22. Prove Lemma 1.7.

[36]This was on author MLP's final exam when he took a Markov decision process course in graduate school many years ago.

23. Write out and prove Lemma 1.4 in component notation.

24. Of the four value functions in Example 1.5, verify that $\mathbf{v}_\lambda^{d_4^\infty}$ is the only one that satisfies the Bellman equation.

25. 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.

26. Prove Theorem 1.22.

27. Referring to the equipment maintenance example from Section 1.9.2, develop an algorithm for finding an optimal policy by searching within the class of control limit policies.

28. Apply Gauss-Seidel iteration directly in Example 1.19 and verify that it gives the same result.

Bibliography

O. Alagoz, L. M. Maillart, A. J. Schaefer, and M. S. Roberts. Determining the acceptance of cadaveric livers using an implicit model of the waiting list. *Operations Research*, 55:24–36, 2007.

K. J. Arrow, S. Karlin, and H. E. Scarf. *Studies in the Mathematical Theory of Inventory and Production.* Stanford University Press, Stanford, CA, 1958.

J.A. Bather. Optimal decision procedures for finite Markov chains, i. *Adv. Appl. Prob.*, 5:328–339, 1973.

R.E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

D.P. Bertseksas. *Dynamic Programming and Optimal Control, Volume 2, 4th Edition.* Athena Scientific, 2012.

D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts, 1997.

D. Blackwell. Discrete dynamic programming. *Ann. Math. Stat.*, 33:719–726, 1962.

D. Blackwell. Discounted dynamic programming. *Ann. Math. Stat.*, 36:226–235, 1965.

P¿G¿ Canbolat and U.G. Rothblum. (approximate)iterated successive approximations algorithm for sequential decision processes. *Ann. Oper. Res.*, 208:309–320, 2013.

F. D'Epenoux. Sur un problème de production et de stockage dans l'aléatoire. *RAIRO*, 14:3–16, 1960.

C. Derman. *Finite State Markovian Decision Processes.* Academic Press, Newyork, NY., 1970.

N.A.J. Hastings. Some notes on dynamic programming and replacement. *Op. Res. Quart.*, 19:453–464, 1968.

F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research.* McGraw-Hill, 2021.

R. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA, 1960.

L.C.M. Kallenberg. *Linear Programming and Finite Markovian Control Problems.* Mathematisch Centrum, Amsterdam, 1983.

H. Kushner and A.J. Kleimnan. Accelerated procedures for the solution of discrete Markov control problems. *IEEE Trans. Automat. Control*, 16:147–152, 1971.

J. MacQueen. A modified dynamic programming method for Markov decision problems. *J. Math. Anal. Appl.*, 14:38–43, 1966.

J. MacQueen. Letter to the editor - a test for suboptimal actions in Markov decision problems. *Operations Research*, 15:559–561, 1967.

E. Porteus. *Foundations of Stochastic Inventory Theory.* Stanford Business Books, 2002.

M. Puterman and S. Brumelle. On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 4:60–69, 1979.

M. L. Puterman. *Markov Decision Processes:Discrete Stochastic Dynamic Programming.* John Wiley & Sons., 1994.

M. L. Puterman and M.C. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Man. Sci.*, 24:1127–1137, 1978.

R. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

L.S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. USA*, 39:1095–1100, 1953.

R. Strauch. Negative dynamic programming. *Ann. Math. Stat.*, 37:871–890, 1966.

J.A.E.E. van Nunen. A set of successive approximation methods for discounted Markovian decision problems. *Z. Op. Res.*, 20:203–208, 1976.