Chapter 3
Dynamic Programming

This chapter introduces basic ideas and methods of dynamic programming.$^1$ It sets out the basic elements of a recursive optimization problem, describes a key functional equation called the Bellman equation, presents three methods for solving the Bellman equation, and gives the Benveniste-Scheinkman formula for the derivative of the optimal value function. Let's dive in.

3.1. Sequential problems

Let $\beta \in (0, 1)$ be a discount factor. We want to choose an infinite sequence of “controls” $\{u_t\}^\infty_{t=0}$ to maximize

$$\sum_{t=0}^{\infty} \beta^t r(x_t, u_t),$$  \hspace{1cm} (3.1.1)

subject to $x_{t+1} = g(x_t, u_t)$, with $x_0 \in \mathbb{R}^n$ given. We assume that $r(x_t, u_t)$ is a concave function and that the set $\{(x_{t+1}, x_t) : x_{t+1} \leq g(x_t, u_t), u_t \in \mathbb{R}^k\}$ is convex and compact. Dynamic programming seeks a time-invariant \textit{policy function} $h$ mapping the state $x_t$ into the control $u_t$, such that the sequence $\{u_t\}^\infty_{t=0}$ generated by iterating the two functions

$$u_t = h(x_t)$$
$$x_{t+1} = g(x_t, u_t),$$  \hspace{1cm} (3.1.2)

starting from initial condition $x_0$ at $t = 0$, solves the original problem. A solution in the form of equations (3.1.2) is said to be \textit{recursive}. To find the policy function $h$ we need to know another function $V(x)$ that expresses the optimal value of the original problem, starting from an arbitrary initial condition $x \in X$. This is called the \textit{value function}. In particular, define

$$V(x_0) = \max_{\{u_t\}^\infty_{t=0}} \sum_{t=0}^{\infty} \beta^t r(x_t, u_t),$$  \hspace{1cm} (3.1.3)

$^1$ This chapter aims to the reader to start using the methods quickly. We hope to promote demand for further and more rigorous study of the subject. In particular see Bertsekas (1976), Bertsekas and Shreve (1978), Stokey and Lucas (with Prescott) (1989), Bellman (1957), and Chow (1981). This chapter covers much of the same material as Sargent (1987b, chapter 1).
where again the maximization is subject to $x_{t+1} = g(x_t, u_t)$, with $x_0$ given. Of course, we cannot possibly expect to know $V(x_0)$ until after we have solved the problem, but let’s proceed on faith. If we knew $V(x_0)$, then the policy function $h$ could be computed by solving for each $x \in X$ the problem

$$\max_u \{ r(x, u) + \beta V(\tilde{x}) \},$$

(3.1.4)

where the maximization is subject to $\tilde{x} = g(x, u)$ with $x$ given, and $\tilde{x}$ denotes the state next period. Thus, we have exchanged the original problem of finding an infinite sequence of controls that maximizes expression (3.1.1) for the problem of finding the optimal value function $V(x)$ and a function $h$ that solves the continuum of maximum problems (3.1.4)—one maximum problem for each value of $x$. This exchange doesn’t look like progress, but we shall see that it often is.

Our task has become jointly to solve for $V(x), h(x)$, which are linked by the Bellman equation

$$V(x) = \max_u \{ r(x, u) + \beta V(g(x, u)) \}. \quad (3.1.5)$$

The maximizer of the right side of equation (3.1.5) is a policy function $h(x)$ that satisfies

$$V(x) = r(x, h(x)) + \beta V(g(x, h(x))). \quad (3.1.6)$$

Equation (3.1.5) or (3.1.6) is a functional equation to be solved for the pair of unknown functions $V(x), h(x)$.

Methods for solving the Bellman equation are based on mathematical structures that vary in their details depending on the precise nature of the functions $r$ and $g$. All of these structures contain versions of the following four findings. Under various particular assumptions about $r$ and $g$, it turns out that

---

2 There are alternative sets of conditions that make the maximization (3.1.4) well behaved. One set of conditions is as follows: (1) $r$ is concave and bounded, and (2) the constraint set generated by $g$ is convex and compact, that is, the set of $\{(x_{t+1}, x_t) : x_{t+1} \leq g(x_t, u_t)\}$ for admissible $u_t$ is convex and compact. See Stokey, Lucas, and Prescott (1989) and Bertsekas (1976) for further details of convergence results. See Benveniste and Scheinkman (1979) and Stokey, Lucas, and Prescott (1989) for the results on differentiability of the value function. In Appendix A (see Technical Appendixes), we describe the mathematics for one standard set of assumptions about $(r, g)$. In chapter 5, we describe it for another set of assumptions about $(r, g)$. 

---
1. The functional equation (3.1.5) has a unique strictly concave solution.

2. This solution is approached in the limit as \( j \to \infty \) by iterations on

\[
V_{j+1}(x) = \max_u \{ r(x, u) + \beta V_j(\tilde{x}) \},
\]

subject to \( \tilde{x} = g(x, u), x \) given, starting from any bounded and continuous initial \( V_0 \).

3. There is a unique and time-invariant optimal policy of the form \( u_t = h(x_t) \),

where \( h \) is chosen to maximize the right side of (3.1.5).

4. Off corners, the limiting value function \( V \) is differentiable.

Since the value function is differentiable, the first-order necessary condition for problem (3.1.4) becomes\(^3\)

\[
r_2(x, u) + \beta V'(g(x, u)) g_2(x, u) = 0. \tag{3.1.7}
\]

If we also assume that the policy function \( h(x) \) is differentiable, differentiation of expression (3.1.6) yields\(^4\)

\[
V'(x) = r_1[x, h(x)] + r_2[x, h(x)] h'(x) + \beta V'(g[x, h(x)]) \left\{ g_1[x, h(x)] + g_2[x, h(x)] h'(x) \right\}. \tag{3.1.8}
\]

When the states and controls can be defined in such a way that only \( u \) appears in the transition equation, i.e., \( \tilde{x} = g(u) \), the derivative of the value function becomes, after substituting expression (3.1.7) with \( u = h(x) \) into (3.1.8),

\[
V'(x) = r_1[x, h(x)]. \tag{3.1.9}
\]

This is a version of a formula of Benveniste and Scheinkman (1979).

At this point, we describe three broad computational strategies that apply in various contexts.

---

\(^3\) Here and below, subscript 1 denotes the vector of derivatives with respect to the \( x \) components and subscript 2 denotes the derivatives with respect to the \( u \) components.

\(^4\) Benveniste and Scheinkman (1979) proved differentiability of \( V(x) \) under broad conditions that do not require that \( h(x) \) be differentiable. For conditions under which \( h(x) \) is differentiable, see Santos (1991,1993).
3.1.1. Three computational methods

There are three main types of computational methods for solving dynamic programs. All aim to solve the functional equation (3.1.4).

**Value function iteration.** The first method proceeds by constructing a sequence of value functions and associated policy functions. The sequence is created by iterating on the following equation, starting from $V_0 = 0$, and continuing until $V_j$ has converged:

$$V_{j+1}(x) = \max_u \{ r(x,u) + \beta V_j(\bar{x}) \},$$

subject to $\bar{x} = g(x,u)$, $x$ given.\(^5\) This method is called value function iteration or iterating on the Bellman equation.

**Guess and verify.** A second method involves guessing and verifying a solution $V$ to equation (3.1.5). This method relies on the uniqueness of the solution to the equation, but because it relies on luck in making a good guess, it is not generally available.

**Howard's improvement algorithm.** A third method, known as policy function iteration or Howard's improvement algorithm, consists of the following steps:

1. Pick a feasible policy, $u = h_0(x)$, and compute the value associated with operating forever with that policy:

   $$V_{h_j}(x) = \sum_{t=0}^{\infty} \beta^t r[x_t, h_j(x_t)],$$

   where $x_{t+1} = g[x_t, h_j(x_t)]$, with $j = 0$.

2. Generate a new policy $u = h_{j+1}(x)$ that solves the two-period problem

   $$\max_u \{ r(x,u) + \beta V_{h_j} [g(x,u)] \},$$

   for each $x$.

---

\(^5\) See Appendix A on functional analysis (see Technical Appendixes) for what it means for a sequence of functions to converge. A proof of the uniform convergence of iterations on equation (3.1.10) is contained in that appendix.
3. Iterate over $j$ to convergence on steps 1 and 2.

In Appendix A (see Technical Appendices), we describe some conditions under which the policy improvement algorithm converges to the solution of the Bellman equation. The policy improvement algorithm often converges faster than does value function iteration (e.g., see exercise 3.1 at the end of this chapter). The policy improvement algorithm is also a building block for methods used to study government policy in chapter 23.

Each of our three methods for solving dynamic programming problems has its uses. Each is easier said than done, because it is typically impossible analytically to compute even one iteration on equation (3.1.10). This fact thrusts us into the domain of computational methods for approximating solutions: pencil and paper are insufficient. Chapter 4 describes computational methods that can applied to problems that cannot be solved by hand. Here we shall describe the first of two special types of problems for which analytical solutions can be obtained. It involves Cobb-Douglas constraints and logarithmic preferences. Later, in chapter 5, we shall describe a specification with linear constraints and quadratic preferences. For that special case, many analytic results are available. These two classes have been important in economics as sources of examples and as inspirations for approximations.

### 3.1.2. Cobb-Douglas transition, logarithmic preferences

Brock and Mirman (1972) used the following optimal growth example. A planner chooses sequences $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ to maximize

$$
\sum_{t=0}^{\infty} \beta^t \ln(c_t)
$$

subject to a given value for $k_0$ and a transition law

$$
k_{t+1} + c_t = Ak_t^\alpha, \quad (3.1.11)
$$

where $A > 0, \alpha \in (0, 1), \beta \in (0, 1)$.

---

6 The speed of the policy improvement algorithm comes from its implementing Newton's method, which converges quadratically while iteration on the Bellman equation converges at a linear rate. See chapter 4 and Appendix A (see Technical Appendices).

7 See also Levhari and Srinivasan (1969).
This problem can be solved “by hand,” using any of our three methods. We begin with iteration on the Bellman equation. Start with $v_0(k) = 0$, and solve the one-period problem: choose $c$ to maximize $\ln(c)$ subject to $c + \tilde{k} = Ak^\alpha$. The solution is evidently to set $c = Ak^\alpha$, $\tilde{k} = 0$, which produces an optimized value $v_1(k) = \ln A + \alpha \ln k$. At the second step, we find $c = \frac{1}{1+\beta\alpha} Ak^\alpha$, $\tilde{k} = \frac{\beta\alpha}{1+\beta\alpha} Ak^\alpha$, $v_2(k) = \ln \frac{A}{1+\alpha\beta} + \beta \ln A + \alpha \beta \ln \frac{\alpha}{1+\alpha\beta} + \alpha(1+\alpha\beta) \ln k$. Continuing, and using the algebra of geometric series, gives the limiting policy functions $c = (1-\beta\alpha)Ak^\alpha$, $\tilde{k} = \beta\alpha Ak^\alpha$, and the value function $v(k) = (1-\beta\alpha)^{-1} [\ln(A(1-\beta\alpha)) + \frac{\beta\alpha}{1-\beta\alpha} \ln k]$. Here is how the guess-and-verify method applies to this problem. Since we already know the answer, we’ll guess a function of the correct form, but leave its coefficients undetermined.\footnote{This is called the method of undetermined coefficients.} Thus, we make the guess

$$v(k) = E + F \ln k,$$

(3.1.12)

where $E$ and $F$ are undetermined constants. The left and right sides of equation (3.1.12) must agree for all values of $k$. For this guess, the first-order necessary condition for the maximum problem on the right side of equation (3.1.10) implies the following formula for the optimal policy $\tilde{k} = h(k)$, where $\tilde{k}$ is next period’s value and $k$ is this period’s value of the capital stock:

$$\tilde{k} = \frac{\beta F}{1+\beta F} Ak^\alpha.$$

(3.1.13)

Substitute equation (3.1.13) into the Bellman equation and equate the result to the right side of equation (3.1.12). Solving the resulting equation for $E$ and $F$ gives $F = \alpha/(1-\alpha\beta)$ and $E = (1-\beta\alpha)^{-1} [\ln A(1-\alpha\beta) + \frac{\beta\alpha}{1-\alpha\beta} \ln A(1-\alpha\beta)]$. It follows that

$$\tilde{k} = \beta\alpha Ak^\alpha.$$

(3.1.14)

Note that the term $F = \alpha/(1-\alpha\beta)$ can be interpreted as a geometric sum $\alpha[1 + \alpha\beta + (\alpha\beta)^2 + \ldots]$. Equation (3.1.14) shows that the optimal policy is to have capital move according to the difference equation $k_{t+1} = A\beta\alpha k_t^\alpha$, or $\ln k_{t+1} = \ln A\beta\alpha + \alpha \ln k_t$. That $\alpha$ is less than 1 implies that $k_t$ converges as $t$ approaches infinity for any positive initial value $k_0$. The stationary point is given by the solution of $k_\infty = A\beta\alpha k_\infty^\alpha$, or $k_\infty^{\alpha-1} = (A\beta\alpha)^{-1}$.\footnote{This is called the method of undetermined coefficients.}
3.1.3. Euler equations

In many problems, there is no unique way of defining states and controls, and several alternative definitions lead to the same solution of the problem. When the states and controls can be defined in such a way that only $u$ appears in the transition equation, i.e., $\tilde{x} = g(u)$: the first-order condition for the problem on the right side of the Bellman equation (expression (3.1.7)) in conjunction with the Benveniste-Scheinkman formula (expression (3.1.9)) implies

$$r_2(x_t, u_t) + \beta r_1(x_{t+1}, u_{t+1}) g'(u_t) = 0, \quad x_{t+1} = g(u_t).$$

The first equation is called an Euler equation. Under circumstances in which the second equation can be inverted to yield $u_t$ as a function of $x_{t+1}$, using the second equation to eliminate $u_t$ from the first equation produces a second-order difference equation in $x_t$, since eliminating $u_{t+1}$ brings in $x_{t+2}$.

3.1.4. A sample Euler equation

As an example of an Euler equation, consider the Ramsey problem of choosing \( \{c_t, k_{t+1}\}_{t=0}^{\infty} \) to maximize $\sum_{t=0}^{\infty} \beta^t u(c_t)$ subject to $c_t + k_{t+1} = f(k_t)$, where $k_0$ is given and the one-period utility function satisfies $u'(c) > 0$, $u''(c) < 0$, $\lim_{c \to 0} u'(c) = \infty$, and where $f'(k) > 0$, $f''(k) < 0$. Let the state be $k$ and the control be $\tilde{k}$, where $\tilde{k}$ denotes next period's value of $k$. Substitute $c = f(k) - \tilde{k}$ into the utility function and express the Bellman equation as

$$v(k) = \max_{{\tilde{k}}} \left\{ u \left[ f(k) - \tilde{k} \right] + \beta v \left( \tilde{k} \right) \right\}. \quad (3.1.15)$$

Application of the Benveniste-Scheinkman formula gives

$$v'(k) = u' \left[f(k) - \tilde{k}\right] f'(k). \quad (3.1.16)$$

Notice that the first-order condition for the maximum problem on the right side of equation (3.1.15) is $-u'[f(k) - \tilde{k}] + \beta v'(\tilde{k}) = 0$, which, using equation (3.1.16), gives

$$u' \left[f(k) - \tilde{k}\right] = \beta u' \left[f(\tilde{k}) - \tilde{k}\right] f'(k). \quad (3.1.17)$$
where $\hat{k}$ denotes the two-period-ahead value of $k$. Equation (3.1.17) can be expressed as

$$1 = \beta' \frac{w'(c_{t+1})}{w'(c_t)} f'(k_{t+1}),$$

an Euler equation that is exploited extensively in the theories of finance, growth, and real business cycles.

### 3.2. Stochastic control problems

We now consider a modification of problem (3.1.1) to permit uncertainty. Essentially, we add some well-placed shocks to the previous nonstochastic problem. So long as the shocks are either independently and identically distributed or Markov, straightforward modifications of the method for handling the nonstochastic problem will work.

Thus, we modify the transition equation and consider the problem of maximizing

$$E_0 \sum_{t=0}^{\infty} \beta^t r(x_t, u_t), \quad 0 < \beta < 1,$$

subject to

$$x_{t+1} = g(x_t, u_t, \epsilon_{t+1}),$$

with $x_0$ known and given at $t = 0$, where $\epsilon_t$ is a sequence of independently and identically distributed random variables with cumulative probability distribution function $\text{prob}\{\epsilon_t \leq y\} = F(y)$ for all $t$; $E_t(y)$ denotes the mathematical expectation of a random variable $y$, given information known at $t$. At time $t$, $x_t$ is assumed to be known, but $x_{t+j}, j \geq 1$ is not known at $t$. That is, $\epsilon_{t+1}$ is realized at $(t + 1)$, after $u_t$ has been chosen at $t$. In problem (3.2.1)–(3.2.2), uncertainty is injected by assuming that $x_t$ follows a random difference equation.

Problem (3.2.1)–(3.2.2) continues to have a recursive structure, stemming jointly from the additive separability of the objective function (3.2.1) in pairs $(x_t, u_t)$ and from the difference equation characterization of the transition law (3.2.2). In particular, controls dated $t$ affect returns $r(x_s, u_s)$ for $s \geq t$ but not earlier. This feature implies that dynamic programming methods remain appropriate.
The problem is to maximize expression (3.2.1) subject to equation (3.2.2) by choice of a “policy” or “contingency plan” \( u_t = h(x_t) \). The Bellman equation (3.1.5) becomes

\[
V(x) = \max_u \{ r(x, u) + \beta E[V[g(x, u, \epsilon)]|x] \},
\]

(3.2.3)

where \( E[V[g(x, u, \epsilon)]|x] = \int V[g(x, u, \epsilon)]dF(\epsilon) \) and where \( V(x) \) is the optimal value of the problem starting from \( x \) at \( t = 0 \). The solution \( V(x) \) of equation (3.2.3) can be computed by iterating on

\[
V_{j+1}(x) = \max_u \{ r(x, u) + \beta E[V_j[g(x, u, \epsilon)]|x] \},
\]

(3.2.4)

starting from any bounded continuous initial \( V_0 \). Under various particular regularity conditions, there obtain versions of the same four properties listed earlier.

The first-order necessary condition for the problem on the right side of equation (3.2.3) is

\[
\begin{align*}
    r_2(x, u) + \beta E \left\{ V'[g(x, u, \epsilon)] g_2(x, u, \epsilon) \bigg| x \right\} = 0,
\end{align*}
\]

which we obtained simply by differentiating the right side of equation (3.2.3), passing the differentiation operation under the \( E \) (an integration) operator. Off corners, the value function satisfies

\[
V'(x) = r_1[x, h(x)] + r_2[x, h(x)] h'(x)
\]

\[
+ \beta E \left\{ V'[g(x, h(x), \epsilon)] \{ g_1[x, h(x), \epsilon] + g_2[x, h(x), \epsilon] h'(x) \} \bigg| x \right\}.
\]

When the states and controls can be defined in such a way that \( x \) does not appear in the transition equation, the formula for \( V'(x) \) becomes

\[
V'(x) = r_1[x, h(x)].
\]

Substituting this formula into the first-order necessary condition for the problem gives the stochastic Euler equation

\[
r_2(x, u) + \beta E \left[ r_1(\tilde{x}, \tilde{u}) g_2(x, u, \epsilon) \bigg| x \right] = 0,
\]

where tildes over \( x \) and \( u \) denote next-period values.

---

9 See Stokey and Lucas (with Prescott) (1989), or the framework presented in Appendix A (see Technical Appendixes).
3.3. Concluding remarks

This chapter has put forward basic tools and findings: the Bellman equation and several approaches to solving it; the Euler equation; and the Beweniste-Scheinkman formula. To appreciate and believe in the power of these tools requires more words and more practice than we have yet supplied. In the next several chapters, we put the basic tools to work in different contexts with particular specification of return and transition equations designed to render the Bellman equation susceptible to further analysis and computation.

Exercise

Exercise 3.1  Howard’s policy iteration algorithm

Consider the Brock-Mirman problem: to maximize

$$E_0 \sum_{t=0}^{\infty} \beta^t \ln c_t,$$

subject to $c_t + k_{t+1} = Ak_t^\alpha \theta_t$. $k_0$ given, $A > 0$, $1 > \alpha > 0$, where $\{\theta_t\}$ is an i.i.d. sequence with $\ln \theta_t$ distributed according to a normal distribution with mean zero and variance $\sigma^2$.

Consider the following algorithm. Guess at a policy of the form $k_{t+1} = h_0(Ak_t^\alpha \theta_t)$ for any constant $h_0 \in (0,1)$. Then form

$$J_0 (k_0, \theta_0) = E_0 \sum_{t=0}^{\infty} \beta^t \ln (Ak_t^\alpha \theta_t - h_0 Ak_t^\alpha \theta_t).$$

Next choose a new policy $h_1$ by maximizing

$$\ln (Ak^\alpha \theta - k') + \beta E J_0 (k', \theta'),$$

where $k' = h_1 Ak^\alpha \theta$. Then form

$$J_1 (k_0, \theta_0) = E_0 \sum_{t=0}^{\infty} \beta^t \ln (Ak_t^\alpha \theta_t - h_1 Ak_t^\alpha \theta_t).$$

Continue iterating on this scheme until successive $h_j$ have converged.

Show that, for the present example, this algorithm converges to the optimal policy function in one step.
Chapter 4
Practical Dynamic Programming

4.1. The curse of dimensionality

We often encounter problems where it is impossible to attain closed forms for iterating on the Bellman equation. Then we have to adopt numerical approximations. This chapter describes two popular methods for obtaining numerical approximations. The first method replaces the original problem with another problem that forces the state vector to live on a finite and discrete grid of points, then applies discrete-state dynamic programming to this problem. The “curse of dimensionality” impels us to keep the number of points in the discrete state space small. The second approach uses polynomials to approximate the value function. Judd (1998) is a comprehensive reference about numerical analysis of dynamic economic models and contains many insights about ways to compute dynamic models.

4.2. Discrete-state dynamic programming

We introduce the method of discretization of the state space in the context of a particular discrete-state version of an optimal savings problem. An infinitely lived household likes to consume one good that it can acquire by spending labor income or accumulated savings. The household has an endowment of labor at time $t$, $s_t$, that evolves according to an $m$-state Markov chain with transition matrix $P$ and state space $[\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_m]$. If the realization of the process at $t$ is $\bar{s}_i$, then at time $t$ the household receives labor income of amount $w\bar{s}_i$. The wage $w$ is fixed over time. We shall sometimes assume that $m$ is 2, and that $s_t$ takes on value 0 in an unemployed state and 1 in an employed state. In this case, $w$ has the interpretation of being the wage of employed workers.

The household can choose to hold a single asset in discrete amounts $a_t \in A$ where $A$ is a grid $[a_1 < a_2 < \cdots < a_n]$. How the model builder chooses the
end points of the grid $\mathcal{A}$ is important, as we describe in detail in chapter 18 on incomplete market models. The asset bears a gross rate of return $r$ that is fixed over time.

The household’s maximum problem, for given values of $(w, r)$ and given initial values $(a_0, s_0)$, is to choose a policy for $\{a_{t+1}\}_{t=0}^{\infty}$ to maximize

$$E \sum_{i=0}^{\infty} \beta^i u(c_i),$$

subject to

$\begin{align*}
  c_t + a_{t+1} &= (r+1)a_t + ws_t \\
  c_t &\geq 0 \\
  a_{t+1} &\in \mathcal{A}
\end{align*}$

where $\beta \in (0,1)$ is a discount factor and $r$ is fixed rate of return on the assets. We assume that $\beta(1 + r) < 1$. Here $u(c)$ is a strictly increasing, concave one-period utility function. Associated with this problem is the Bellman equation

$$v(a, s) = \max_{a' \in \mathcal{A}} \left\{ u[(r+1)a + ws - a'] + \beta Ev(a', s') \right\},$$

where $a$ is next period’s value of asset holdings, and $s'$ is next period’s value of the shock; here $v(a, s)$ is the optimal value of the objective function, starting from asset, employment state $(a, s)$. We seek a value function $v(a, s)$ that satisfies equation (18.2.3) and an associated policy function $a' = g(a, s)$ mapping this period’s $(a, s)$ pair into an optimal choice of assets to carry into next period. Let assets live on the grid $\mathcal{A} = [a_1, a_2, \ldots, a_n]$. Then we can express the Bellman equation as

$$v(a_i, s_j) = \max_{a_h \in \mathcal{A}} \left\{ u[(r+1)a_i + ws_j - a_h] + \beta \sum_{i=1}^{m} P_{ji} v(a_h, s_i) \right\},$$

for each $i \in \{1, \ldots, n\}$ and each $j \in \{1, \ldots, m\}$. 
4.3. Bookkeeping

For a discrete state space of small size, it is easy to solve the Bellman equation numerically by manipulating matrices. Here is how to write a computer program to iterate on the Bellman equation in the context of the preceding model of asset accumulation.\(^1\) Let there be \(n\) states \([a_1, a_2, \ldots, a_n]\) for assets and two states \([s_1, s_2]\) for employment status. For \(j = 1, 2\), define \(n \times 1\) vectors \(v_j, j = 1, 2\), whose \(i\)th rows are determined by \(v_j(i) = v(a_i, s_j), i = 1, \ldots, n\). Let \(1\) be the \(n \times 1\) vector consisting entirely of ones. For \(j = 1, 2\), define two \(n \times n\) matrices \(R_j\) whose \((i, h)\) elements are

\[
R_j(i, h) = u[(r + 1)a_i + ws_j - ah], \quad i = 1, \ldots, n, h = 1, \ldots, n.
\]

Define an operator \(T([v_1, v_2])\) that maps a pair of \(n \times 1\) vectors \([v_1, v_2]\) into a pair of \(n \times 1\) vectors \([tv_1, tv_2]\):\(^2\)

\[
tv_j(i) = \max_h \left\{ R_j(i, h) + \beta P_{ji} v_1(h) + \beta P_{j2} v_2(h) \right\}
\]

for \(j = 1, 2\), or

\[
tv_1 = \max\{R_1 + \beta P_{11} v_1 + \beta P_{12} v_2\}
\]

\[
tv_2 = \max\{R_2 + \beta P_{21} v_1 + \beta P_{22} v_2\}.
\]

(4.3.1)

Here it is understood that the “max” operator applied to an \((n \times m)\) matrix \(M\) returns an \((n \times 1)\) vector whose \(i\)th element is the maximum of the \(i\)th row of the matrix \(M\). These two equations can be written compactly as

\[
\begin{bmatrix}
tv_1 \\
tv_2
\end{bmatrix} = \max \left\{ \begin{bmatrix}
R_1 \\
R_2
\end{bmatrix} + \beta (P \otimes 1) \begin{bmatrix}
v_1 \\
v_2
\end{bmatrix} \right\},
\]

(4.3.2)

where \(\otimes\) is the Kronecker product.\(^3\)

\(^1\) Matlab versions of the program have been written by Gary Hansen, Selahattin Imrohoroglu, George Hall, and Chao Wei.

\(^2\) Programming languages like Gauss and Matlab execute maximum operations over vectors very efficiently. For example, for an \(n \times m\) matrix \(A\), the Matlab command \([r, \text{index}] = \max(A)\) returns the two \((1 \times m)\) row vectors \(r, \text{index}\), where \(r_j = \max_i A(i, j)\) and \(\text{index}_j\) is the row \(i\) that attains \(\max_i A(i, j)\) for column \(j\) \([\text{i.e., } \text{index}_j = \arg\max_i A(i, j)\)]). This command performs \(m\) maximizations simultaneously.

\(^3\) If \(A\) is an \(m\)-by-\(n\) matrix and \(B\) is a \(p\)-by-\(q\) matrix, then the Kronecker product \(A \otimes B\) is the \(mp\)-by-\(nq\) block matrix

\[
A \otimes B = \begin{bmatrix}
a_{11}B & \cdots & a_{1n}B \\
\vdots & \ddots & \vdots \\
a_{m1}B & \cdots & a_{mn}B
\end{bmatrix}.
\]
The Bellman equation \([v_1, v_2] = T([v_1, v_2])\) can be solved by iterating to convergence on \([v_1, v_2]_{m+1} = T([v_1, v_2]_m)\).

### 4.4. Application of Howard improvement algorithm

Often computation speed is important. Exercise 3.1 showed that the policy improvement algorithm can be much faster than iterating on the Bellman equation. It is also easy to implement the Howard improvement algorithm in the present setting. At time \(t\), the system resides in one of \(N\) predetermined positions, denoted \(x_i\) for \(i = 1, 2, \ldots, N\). There exists a predetermined set \(\mathcal{M}\) of \((N \times N)\) stochastic matrices \(P\) that are the objects of choice. Here \(P_{ij} = \text{Prob}[x_{t+1} = x_j \mid x_t = x_i]\), \(i = 1, \ldots, N; j = 1, \ldots, N\).

The matrices \(P\) satisfy \(P_{ij} \geq 0\), \(\sum_{j=1}^{N} P_{ij} = 1\), and additional restrictions dictated by the problem at hand that determine the set \(\mathcal{M}\). The one-period return function is represented as \(c_P\), a vector of length \(N\), and is a function of \(P\). The \(i\)th entry of \(c_P\) denotes the one-period return when the state of the system is \(x_i\) and the transition matrix is \(P\). The Bellman equation is

\[
v_P(x_i) = \max_{P \in \mathcal{M}} \{c_P(x_i) + \beta \sum_{j=1}^{N} P_{ij} v_P(x_j)\}
\]

or

\[
v_P = \max_{P \in \mathcal{M}} \{c_P + \beta P v_P\}.
\]  

(4.4.1)

We can express this as

\[
v_P = T v_P,
\]

where \(T\) is the operator defined by the right side of (4.4.1). Following Puterman and Brumelle (1979) and Puterman and Shin (1978), define the operator

\[
B = T - I,
\]

so that

\[
Bv = \max_{P \in \mathcal{M}} \{c_P + \beta P v\} - v.
\]

In terms of the operator \(B\), the Bellman equation is

\[
Bv = 0.
\]  

(4.4.2)
The policy improvement algorithm consists of iterations on the following two steps.

1. For fixed $P_n$, solve
   \[(I - \beta P_n) v_{P_n} = c_{P_n}\]  
   \[(4.4.3)\]
   for $v_{P_n}$.

2. Find $P_{n+1}$ such that
   \[c_{P_{n+1}} + (\beta P_{n+1} - I) v_{P_n} = B v_{P_n}\]
   \[(4.4.4)\]
   Step 1 is accomplished by setting
   \[v_{P_n} = (I - \beta P_n)^{-1} c_{P_n}.\]
   \[(4.4.5)\]
   Step 2 amounts to finding a policy function (i.e., a stochastic matrix $P_{n+1} \in M$) that solves a two-period problem with $v_{P_n}$ as the terminal value function.

Following Puterman and Brumelle, the policy improvement algorithm can be interpreted as a version of Newton’s method for finding the zero of $Bv = v$. Using equation (4.4.3) for $n+1$ to eliminate $c_{P_{n+1}}$ from equation (4.4.4) gives

\[(I - \beta P_{n+1}) v_{P_{n+1}} + (\beta P_{n+1} - I) v_{P_n} = B v_{P_n}\]
\[(4.4.6)\]
which implies
\[v_{P_{n+1}} = v_{P_n} + (I - \beta P_{n+1})^{-1} B v_{P_n}.\]

From equation (4.4.4), $(\beta P_{n+1} - I)$ can be regarded as the gradient of $Bv_{P_n}$, which supports the interpretation of equation (4.4.6) as implementing Newton’s method.\(^4\)

---

\(^4\) Newton’s method for finding the solution of $G(z) = 0$ is to iterate on $z_{n+1} = z_n - G'(z_n)^{-1} G(z_n)$. 
4.5. Numerical implementation

We shall illustrate Howard’s policy improvement algorithm by applying it to our savings example. Consider a feasible policy function \( a' = g(k, s) \). For each \( j \), define the \( n \times n \) matrices \( J_j \) by

\[
J_j(a, a') = \begin{cases} 
1 & \text{if } g(a, s_j) = a' \\
0 & \text{otherwise} 
\end{cases}
\]

Here \( j = 1, 2, \ldots, m \) where \( m \) is the number of possible values for \( s_t \), and \( J_j(a, a') \) is the element of \( J_j \) with rows corresponding to initial assets \( a \) and columns to terminal assets \( a' \). For a given policy function \( a' = g(a, s) \) define the \( n \times 1 \) vectors \( r_j \) with rows corresponding to

\[
r_j(a) = u \left[ (r + 1) a + w s_j - g(a, s_j) \right],
\]

for \( j = 1, \ldots, m \).

Suppose the policy function \( a' = g(a, s) \) is used forever. Let the value associated with using \( g(a, s) \) forever be represented by the \( m \) \((n \times 1)\) vectors \([v_1, \ldots, v_m]\), where \( v_j(a_i) \) is the value starting from state \((a_i, s_j)\). Suppose that \( m = 2 \). The vectors \([v_1, v_2]\) obey

\[
\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} + \begin{bmatrix} \beta P_{11} J_1 & \beta P_{12} J_1 \\ \beta P_{21} J_2 & \beta P_{22} J_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}.
\]

Then

\[
\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \left[ I - \beta \begin{bmatrix} P_{11} J_1 & P_{12} J_1 \\ P_{21} J_2 & P_{22} J_2 \end{bmatrix} \right]^{-1} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}.
\]

Here is how to implement the Howard policy improvement algorithm.

Step 1. For an initial feasible policy function \( g_0(a, j) \) for \( \tau = 1 \), form the \( r_j \) matrices using equation (4.5.1), then use equation (4.5.2) to evaluate the vectors of values \([v_1^*, v_2^*]\) implied by using that policy forever.

Step 2. Use \([v_1^*, v_2^*]\) as the terminal value vectors in equation (4.3.2), and perform one step on the Bellman equation to find a new policy function \( g_{\tau+1}(a, s) \) for \( \tau + 1 = 2 \). Use this policy function, increment \( \tau \) by 1, and repeat step 1.

Step 3. Iterate to convergence on steps 1 and 2.
4.5.1. Modified policy iteration

Researchers have had success using the following modification of policy iteration: for \( k \geq 2 \), iterate \( k \) times on the Bellman equation. Take the resulting policy function and use equation (4.5.2) to produce a new candidate value function. Then starting from this terminal value function, perform another \( k \) iterations on the Bellman equation. Continue in this fashion until the decision rule converges.

4.6. Sample Bellman equations

This section presents some examples. The first two examples involve no optimization, just computing discounted expected utility. Appendix A of chapter 6 describes some related examples based on search theory.

4.6.1. Example 1: calculating expected utility

Suppose that the one-period utility function is the constant relative risk aversion form \( u(c) = c^{1-\gamma}/(1-\gamma) \). Suppose that \( c_{t+1} = \lambda_{t+1} c_t \) and that \( \{\lambda_t\} \) is an \( n \)-state Markov process with transition matrix \( P_{ij} = \text{Prob}(\lambda_{t+1} = \lambda_j | \lambda_t = \lambda_i) \). Suppose that we want to evaluate discounted expected utility

\[
V(c_0, \lambda_0) = E_0 \sum_{t=0}^{\infty} \beta^t u(c_t),
\]

where \( \beta \in (0,1) \). We can express this equation recursively:

\[
V(c_t, \lambda_t) = u(c_t) + \beta E_t V(c_{t+1}, \lambda_{t+1})
\]

We use a guess-and-verify technique to solve equation (4.6.2) for \( V(c_t, \lambda_t) \). Guess that \( V(c_t, \lambda_t) = u(c_t) w(\lambda_t) \) for some function \( w(\lambda_t) \). Substitute the guess into equation (4.6.2), divide both sides by \( u(c_t) \), and rearrange to get

\[
w(\lambda_t) = 1 + \beta E_t \left( \frac{c_{t+1}}{c_t} \right)^{1-\gamma} w(\lambda_{t+1})
\]

or

\[
w_t = 1 + \beta \sum_j P_{ij} (\lambda_j)^{1-\gamma} w_j.
\]
Equation (4.6.3) is a system of linear equations in $w_i, i = 1, \ldots, n$ whose solution can be expressed as

$$w = \left[1 - \beta P \text{ diag} \left(\lambda_1^{1-\gamma}, \ldots, \lambda_n^{1-\gamma}\right)\right]^{-1} \mathbf{1}$$

where $\mathbf{1}$ is an $n \times 1$ vector of ones.

### 4.6.2. Example 2: risk-sensitive preferences

Suppose we modify the preferences of the previous example to be of the recursive form

$$V(c_t, \lambda_t) = u(c_t) + \beta R_t V(c_{t+1}, \lambda_{t+1}), \quad (4.6.4)$$

where

$$R_t (V) = \left(\frac{2}{\sigma}\right) \log E_t \left[\exp \left(\frac{\sigma V_{t+1}}{2}\right)\right] \quad (4.6.5)$$

is an operator used by Jacobson (1973), Whittle (1990), and Hansen and Sargent (1995) to induce a preference for robustness to model misspecification.\footnote{Also see Epstein and Zin (1989) and Weil (1989) for a version of the $R_t$ operator.} Here $\sigma \leq 0$; when $\sigma < 0$, it represents a concern for model misspecification, or an extra sensitivity to risk.

We leave it to the reader to propose a method for computing an approximation to a value function that solves the functional equation (4.6.4). (Hint: the method used in example 1 will not apply directly because the homogeneity property exploited there fails to prevail now.)
4.6.3. Example 3: costs of business cycles

Robert E. Lucas, Jr., (1987) proposed that the cost of business cycles be measured in terms of a proportional upward shift in the consumption process that would be required to make a representative consumer indifferent between its random consumption allocation and a nonrandom consumption allocation with the same mean. This measure of business cycles is the fraction $\Omega$ that satisfies

$$E_0 \sum_{t=0}^{\infty} \beta^t u[(1 + \Omega) c_t] = \sum_{t=0}^{\infty} \beta^t u[ E_0 (c_t)] . \tag{4.6.6}$$

Suppose that the utility function and the consumption process are as in example 1. Then for given $\Omega$, the calculations in example 1 can be used to calculate the left side of equation (4.6.6). In particular, the left side just equals $u[(1 + \Omega)c_0]w(\lambda)$, where $w(\lambda)$ is calculated from equation (4.6.3). To calculate the right side, we have to evaluate

$$E_0 c_t = c_0 \sum_{\lambda_t, \ldots, \lambda_1} \lambda_t \lambda_{t-1} \cdots \lambda_1 \pi(\lambda_t | \lambda_{t-1}) \pi(\lambda_{t-1} | \lambda_{t-2}) \cdots \pi(\lambda_1 | \lambda_0) , \tag{4.6.7}$$

where the summation is over all possible paths of growth rates between 0 and $t$. In the case of i.i.d. $\lambda_t$, this expression simplifies to

$$E_0 c_t = c_0 (E \lambda)^t , \tag{4.6.8}$$

where $E \lambda_t$ is the unconditional mean of $\lambda$. Under equation (4.6.8), the right side of equation (4.6.6) is easy to evaluate.

Given $\gamma, \pi$, a procedure for constructing the cost of cycles—more precisely, the costs of deviations from mean trend—to the representative consumer is first to compute the right side of equation (4.6.6). Then we solve the following equation for $\Omega$:

$$u[(1 + \Omega)c_0]w(\lambda_0) = \sum_{t=0}^{\infty} \beta^t u[ E_0 (c_t)] .$$

Using a closely related but somewhat different stochastic specification, Lucas (1987) calculated $\Omega$. He assumed that the endowment is a geometric trend with growth rate $\mu$ plus an i.i.d. shock with mean zero and variance $\sigma_z^2$. Starting from a base $\mu = \mu_0$, he found $\mu, \sigma_z$ pairs to which the household is indifferent,
assuming various values of $\gamma$ that he judged to be within a reasonable range.\footnote{See chapter 14 for a discussion of reasonable values of $\gamma$. See Table 1 of Manelli and Sargent (1988) for a correction to Lucas’s calculations.} Lucas found that for reasonable values of $\gamma$, it takes a very small adjustment in the trend rate of growth $\mu$ to compensate for even a substantial increase in the “cyclical noise” $\sigma_{z}$, which meant to him that the costs of business cycle fluctuations are small.

Subsequent researchers have studied how other preference specifications would affect the calculated costs. Tallarini (1996, 2000) used a version of the preferences described in example 2 and found larger costs of business cycles when parameters are calibrated to match data on asset prices. Hansen, Sargent, and Tallarini (1999) and Alvarez and Jermann (1999) considered local measures of the cost of business cycles and provided ways to link them to the equity premium puzzle, to be studied in chapter 14.

4.7. Polynomial approximations

Judd (1998) describes a method for iterating on the Bellman equation using a polynomial to approximate the value function and a numerical optimizer to perform the optimization at each iteration. We describe this method in the context of the Bellman equation for a particular problem that we shall encounter later.

In chapter 20, we shall study Hopenhayn and Nicolini’s (1997) model of optimal unemployment insurance. A planner wants to provide incentives to an unemployed worker to search for a new job while also partially insuring the worker against bad luck in the search process. The planner seeks to deliver discounted expected utility $V$ to an unemployed worker at minimum cost while providing proper incentives to search for work. Hopenhayn and Nicolini show that the minimum cost $C(V)$ satisfies the Bellman equation

$$C(V) = \min_{V'} \{ c + \beta [1 - p(a)] C(V') \} \quad (4.7.1)$$

where $c, a$ are given by

$$c = u^{-1} \left( \max \{ 0, V + a - \beta (p(a) V^e + [1 - p(a)] V^u) \} \right). \quad (4.7.2)$$
and
\[
 a = \max \left\{ 0, \frac{\log [r \beta (V^u - V^d)]}{r} \right\}.
\]  

(4.7.3)

Here $V$ is a discounted present value that an insurer has promised to an unemployed worker, $V^u$ is a value for next period that the insurer promises the worker if he remains unemployed, $1 - p(a)$ is the probability of remaining unemployed if the worker exerts search effort $a$, and $c$ is the worker’s consumption level. Hopenhayn and Nicolini assume that $p(a) = 1 - \exp(ra)$, $r > 0$.

### 4.7.1. Recommended computational strategy

To approximate the solution of the Bellman equation (4.7.1), we apply a computational procedure described by Judd (1996, 1998). The method uses a polynomial to approximate the $i$th iterate $C_i(V)$ of $C(V)$. This polynomial is stored on the computer in terms of $n + 1$ coefficients. Then at each iteration, the Bellman equation is to be solved at a small number $m \geq n + 1$ values of $V$. This procedure gives values of the $i$th iterate of the value function $C_i(V)$ at those particular $V$’s. Then we interpolate (or “connect the dots”) to fill in the continuous function $C_i(V)$. Substituting this approximation $C_i(V)$ for $C(V)$ in equation (4.7.1), we pass the minimum problem on the right side of equation (4.7.1) to a numerical minimizer. Programming languages like Matlab and Gauss have easy-to-use algorithms for minimizing continuous functions of several variables. We solve one such numerical problem minimization for each node value for $V$. Doing so yields optimized value $C_{i+1}(V)$ at those node points. We then interpolate to build up $C_{i+1}(V)$. We iterate on this scheme to convergence. Before summarizing the algorithm, we provide a brief description of Chebyshev polynomials.
4.7.2. Chebyshev polynomials

Where \( n \) is a nonnegative integer and \( x \in \mathbb{R} \), the \( n \)th Chebyshev polynomial, is

\[
T_n(x) = \cos \left( n \cos^{-1} x \right). \tag{4.7.4}
\]

Given coefficients \( c_j, j = 0, \ldots, n \), the \( n \)th-order Chebyshev polynomial approximator is

\[
C_n(x) = c_0 + \sum_{j=1}^{n} c_j T_j(x). \tag{4.7.5}
\]

We are given a real-valued function \( f \) of a single variable \( x \in [-1,1] \). For computational purposes, we want to form an approximator to \( f \) of the form (4.7.5). Note that we can store this approximator simply as the \( n+1 \) coefficients \( c_j, j = 0, \ldots, n \). To form the approximator, we evaluate \( f(x) \) at \( n+1 \) carefully chosen points, then use a least-squares formula to form the \( c_j \)'s in equation (4.7.5). Thus, to interpolate a function of a single variable \( x \) with domain \( x \in [-1,1] \), Judd (1996, 1998) recommends evaluating the function at the \( m \geq n+1 \) points \( x_k, k = 1, \ldots, m \), where

\[
x_k = \cos \left( \frac{2k-1}{2m} \pi \right), \quad k = 1, \ldots, m. \tag{4.7.6}
\]

Here \( x_k \) is the zero of the \( k \)th Chebyshev polynomial on \([-1,1]\). Given the \( m \geq n+1 \) values of \( f(x_k) \) for \( k = 1, \ldots, m \), choose the least-squares values of \( c_j \) as

\[
c_j = \frac{\sum_{k=1}^{m} f(x_k) T_j(x_k)}{\sum_{k=1}^{m} T_j^2(x_k)}, \quad j = 0, \ldots, n \tag{4.7.7}
\]
4.7.3. Algorithm: summary

In summary, applied to the Hopenhayn-Nicolini model, the numerical procedure consists of the following steps:

1. Choose upper and lower bounds for $V^u$, so that $V$ and $V^u$ will be understood to reside in the interval $[\underline{V}^u, \overline{V}^u]$. In particular, set $\overline{V}^u = V^e - \frac{1}{\beta_0}$. The bound required to assure positive search effort, computed in chapter 20. Set $\overline{V}^u = V_{\text{max}}$.

2. Choose a degree $n$ for the approximator, a Chebyshev polynomial, and a number $m \geq n + 1$ of nodes or grid points.

3. Generate the $m$ zeros of the Chebyshev polynomial on the set $[1, -1]$, given by (4.7.6).

4. By a change of scale, transform the $z_i$'s to corresponding points $V^u_\ell$ in $[\underline{V}^u, \overline{V}^u]$.

5. Choose initial values of the $n + 1$ coefficients in the Chebyshev polynomial, for example, $c_j = 0, \ldots, n$. Use these coefficients to define the function $C_i(V^u)$ for iteration number $i = 0$.

6. Compute the function $\tilde{C}_i(V^u) = c + \beta(1 - p(a))C_i(V^u)$, where $c, a$ are determined as functions of $(V, V^u)$ from equations (4.7.2) and (4.7.3). This computation builds in the functional forms and parameters of $u(c)$ and $p(a)$, as well as $\beta$.

7. For each point $V^u_\ell$, use a numerical minimization program to find $C_{i+1}(V^u_\ell) = \min_{V^u} \tilde{C}_i(V^u)$.

8. Using these $m$ values of $C_{i+1}(V^u_\ell)$, compute new values of the coefficients in the Chebyshev polynomials by using “least squares” [formula (4.7.7)]. Return to step 5 and iterate to convergence.
4.7.4. Shape-preserving splines

Judd (1998) points out that because they do not preserve concavity, using Chebyshev polynomials to approximate value functions can cause problems. He recommends the Schumaker quadratic shape-preserving spline. It ensures that the objective in the maximization step of iterating on a Bellman equation will be concave and differentiable (Judd, 1998, p. 441). Using Schumaker splines avoids the type of internodal oscillations associated with other polynomial approximation methods. The exact interpolation procedure is described in Judd (1998, p. 233). A relatively small number of nodes usually is sufficient. Judd and Solnick (1994) find that this approach outperforms linear interpolation and discrete-state approximation methods in a deterministic optimal growth problem.\footnote{The Matlab program schumaker.m (written by Leonardo Rezende of the University of Illinois) can be used to compute the spline. Use the Matlab command \texttt{ppval} to evaluate the spline.}

4.8. Concluding remarks

This chapter has described two of three standard methods for approximating solutions of dynamic programs numerically: discretizing the state space and using polynomials to approximate the value function. The next chapter describes the third method: making the problem have a quadratic return function and linear transition law. A benefit of making the restrictive linear-quadratic assumptions is that they make solving a dynamic program easy by exploiting the ease with which stochastic linear difference equations can be manipulated.