Chapter 7

Discrete Time Discrete State Dynamic Models

With this chapter, we begin our study of dynamic economic models. Dynamic economic models often present three complications rarely encountered together in dynamic physical science models. First, humans are cogent, future-regarding beings capable of assessing how their actions will affect them in the future as well as in the present. Thus, most useful dynamic economic models are future-looking. Second, many aspects of human behavior are unpredictable. Thus, most useful dynamic economic models are inherently stochastic. Third, the predictable component of human behavior is often complex. Thus, most useful dynamic economic models are inherently nonlinear.

The complications inherent in forward-looking, stochastic, nonlinear models make it impossible to obtain explicit analytic solutions to all but a small number of dynamic economic models. However, the proliferation of affordable personal computers, the phenomenal increase of computational speed, and developments of theoretical insights into the efficient use of computers over the last two decades now make it possible for economists to analyze dynamic models much more thoroughly using numerical methods.

The next three chapters are devoted to the numerical analysis of dynamic economic models in discrete time and are followed by three chapters on dynamic economic models in continuous time. In this chapter we study the simplest of these models: the discrete time, discrete state Markov decision model. Though the model is simple, the methods used to analyze the model lay the foundations for the methods developed in subsequent chapters to analyze more complicated models with continuous states and time.

7.1 Discrete Dynamic Programming

The discrete time, discrete state Markov decision model has the following structure: in every period t, an agent observes the state of an economic process s_t, takes an action x_t, and earns a reward $f(x_t, s_t)$ that depends on both the state of the process and the action taken. The state space S, which enumerates all the states attainable by the process, and the action space X, which enumerates all actions that may be taken by the agent, are both finite. The state of the economic process follows a controlled Markov probability law. That is, the distribution of next period's state, conditional on all currently available information, depends only on the current state of the process and the agent's action:

$$\Pr(s_{t+1} = s' | x_t = x, s_t = s, \text{ other information at } t\ ) = P(s'|x, s).$$

The agent seeks a policy $\{x_t^*\}_{t=1}^T$ that prescribes the action $x_t = x_t^*(s_t)$ that should be taken in each state at each point in time so as to maximize the present value of current and expected future rewards over time, discounted at a per-period factor $\delta \in (0, 1]$:

$$\max_{\{x_t^*\}_{t=0}^T} E\left[\sum_{t=0}^{T} \delta^t f(x_t, s_t)\right].$$

A discrete Markov decision model may have an infinite horizon ($T = \infty$) or a finite horizon ($T < \infty$). The model may also be either deterministic or stochastic. It is deterministic if next period's state is known with certainty once the current period's state and action are known. In this case, it is beneficial to dispense with the probability transition law as a description of how the state evolves and use instead a deterministic state transition function g, which explicitly gives the state transitions:

$$s_{t+1} = g(x_t, s_t).$$

Discrete Markov decision models may be analyzed and understood using the dynamic programming principles developed by Richard Bellman (1956). Dynamic programming is an analytic approach in which a multiperiod model is effectively decomposed into a sequence two period models. Dynamic programming is based on the Principle of Optimality, which was articulated by Bellman as follows:

> "An optimal policy has the property that, whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

The Principle of Optimality can be formally expressed in terms of the value functions V_t. For each period t and state s, $V_t(s)$ specifies the maximum attainable sum of current and expected future rewards, given that the process is in state s and the current period is t. Bellman's Principle implies that the value functions must satisfy Bellman's recursion equation

$$V_t(s) = \max_{x \in X(s)} \{f(x,s) + \delta \sum_{s' \in S} P(s'|x,s)V_{t+1}(s')\} \quad s \in S.$$

Bellman's equation captures the essential problem faced by a dynamic, future-regarding optimizing agent: the need to balance the immediate reward $f(x_t, s_t)$ with expected present value of future rewards $\delta E_t V_{t+1}(s_{t+1})$. Given the value functions, the optimal policies $x_t^*(s)$ are simply the solutions to the optimization problems embedded in Bellman's equation.

In a finite horizon model, we adopt the convention that the optimizing agent faces decisions up to and including a final decision period $T < \infty$. The agent faces no decisions after the terminal period T, but may earn a final reward $V_{T+1}(s_{T+1})$ in the subsequent period that depends on the realization of the state in that period. The terminal value is typically fixed by some economically relevant terminal condition. In many applications, V_{T+1} is identically zero, indicating that no rewards are earned by the agent beyond the terminal decision period. In other applications, V_{T+1} may specify a salvage value earned by the agent after making his final decision in period T.

For the finite horizon discrete Markov decision model to be well posed, the terminal value V_{T+1} must be specified by the analyst. Given the terminal value function, the finite horizon decision model in principle may be solved recursively by repeated application of Bellman's equation: having V_{T+1}, solve for $V_T(s)$ for all states s; having V_T, solve for $V_{T-1}(s)$ for all states s; having V_{T-1}, solve for $V_{T-2}(s)$ for all states s; and so on. The process continues until $V_0(s)$ is derived for all states s. Because only finitely many actions are possible, the optimization problem embedded in Bellman's equation can always be solved by performing finitely many arithmetic operations. Thus, the value functions of a finite horizon discrete Markov decision model are always well-defined, although in some cases more than one policy of state-contingent actions may yield the maximum expected stream of rewards, that is, the optimal action may not be unique.

If the decision problem has an infinite horizon, the value functions will not depend on time t. We may, therefore, disregard the time subscripts and denote the common value function by V. Bellman's equation therefore becomes the vector fixed-point equation

$$V(s) = \max_{x \in X(s)} \left[f(x,s) + \delta \sum_{s' \in S} P(s'|x,s)V(s') \right], \quad s \in S.$$

If the discount factor δ is less than one, the mapping underlying Bellman's equation is a strong contraction. The Contraction Mapping Theorem thus guarantees the existence and uniqueness of the infinite horizon value function.[1]

7.2 Economic Examples

Specification of a discrete Markov decision model requires several pieces of information: the state space, the action space, the reward function, the state transition function or state transition probabilities, the discount factor δ, the time horizon T, and, if the model has finite horizon, the terminal value V_{T+1}. This section provides seven economic examples that illustrate how the necessary information is specified and how the Bellman equation is formulated.

7.2.1 Mine Management

A mine operator must determine the optimal ore extraction schedule for a mine that will be shut down and abandoned after T years of operation. The price of extracted ore is p dollars per ton and the total cost of extracting x tons of ore in any year is $c = x^2/(1+s)$ dollars, where s is the tons of ore remaining in the mine at the beginning of the year. The mine currently contains $\bar{s}$ tons of ore. Assuming the amount of ore extracted in any year must be an integer number of tons, what extraction schedule maximizes profits?

This is a finite horizon, deterministic model with time $t = \{1, 2, \ldots, T\}$ measured in years. The state variable

$$s \in S = \{0, 1, 2, \ldots, \bar{s}\}$$

denotes tons of ore remaining in the mine at the beginning of the year. The action variable

$$x \in X(s) = \{0, 1, 2, \ldots, s\}$$

denotes tons of ore extracted over the year. The state transition function is

$$s' = g(s, x) = s - x.$$

The reward function is

$$f(s, x) = px - x^2/(1 + s).$$

[1]Value functions in infinite horizon problems could be time dependent if f, P, or δ displayed time dependence. However, this creates difficulties in developing solution methods, and we have chosen not to explicitly consider this possibility. Fortunately, most infinite horizon economic model do not display such time dependence.

The value of the mine, given it contains s tons of ore at the beginning of year t, satisfies Bellman's equation

$$V_t(s) = \max_{x\in\{0,1,2,\ldots,s\}} \{px - x^2/(1+s) + \delta V_{t+1}(s-x)\}, \qquad s \in S$$

subject to the terminal condition

$$V_{T+1}(s) = 0, \qquad s \in S.$$

7.2.2 Asset Replacement - I

At the beginning of each year, a manufacturer must decide whether to continue operating with an aging physical asset or replace it with a new one. An asset that is a years old yields a profit contribution $p(a)$ up to n years, after which the asset becomes unsafe and must be replaced by law. The cost of a new asset is c. What replacement policy maximizes profits?

This is an infinite horizon, deterministic model with time $t = \{1, 2, \ldots, T\}$ measured in years. The state variable

$$a \in A = \{1, 2, \ldots, n\}$$

denotes the age of the asset in years. The action variable

$$x \in X(a) = \begin{cases} \{\text{keep}, \text{replace}\} & a < n \\ \{\text{replace}\} & a = n \end{cases}$$

denotes the keep-replacement decision. The state transition function is

$$a' = g(a, x) = \begin{cases} a+1 & x = \text{keep} \\ 1 & x = \text{replace}. \end{cases}$$

The reward function is

$$f(a, x) = \begin{cases} p(a) & x = \text{keep} \\ p(0) - c & x = \text{replace}. \end{cases}$$

The value of an asset of age a satisfies Bellman's equation

$$V(a) = \max\{p(a) + \delta V(a+1), p(0) - c + \delta V(1)\}.$$

Bellman's equation asserts that if the manufacturer keeps an asset of age a, he earns $p(a)$ over the coming year and begins the subsequent year with an asset worth $V(a+1)$; if he replaces the asset, on the other hand, he earns $p(0) - c$ over the coming year and begins the subsequent year with an asset worth $V(1)$. Actually, our language is a little loose here. The value $V(a)$ measures not only the current and future net earnings of an asset of age a, but also the net earnings of all future assets that replace it.

7.2.3 Asset Replacement - II

Consider the preceding example, but suppose that the productivity of the asset may be enhanced by performing annual service maintenance. Specifically, at the beginning of each year, a manufacturer must decide whether to replace the asset with a new one or, if he elects to keep the old one, whether to service it. An asset that is a years old and has been serviced s times yields a profit contribution $p(a, s)$ up to and age of n years, after which the asset becomes unsafe and must be replaced by law. The cost of a new asset is c and the cost of servicing an existing asset is k. What replacement policy maximizes profits?

This is an infinite horizon, deterministic model with time $t = \{1, 2, \ldots, T\}$ measured in years. The state variables

$$\begin{aligned} a &\in A = \{1, 2, \ldots, n\} \\ s &\in S = \{0, 1, \ldots, n\} \end{aligned}$$

denote the age of the asset in years and the number of servicings it has undergone, respectively. The action variable

$$x \in X(a, s) = \begin{cases} \{\text{replace, service, no action}\} & a < n \\ \{\text{replace}\} & a = n. \end{cases};$$

The state transition function is

$$(a', s') = g(a, s, x) = \begin{cases} (1, 0) & x = \text{replace} \\ (a+1, s+1) & x = \text{service} \\ (a+1, s) & x = \text{no action.} \end{cases}$$

The reward function is

$$f(a, s, x) = \begin{cases} p(0, 0) - c & x = \text{replace} \\ p(a, s+1) - k & x = \text{service} \\ p(a, s) & x = \text{no action.} \end{cases}$$

The value of asset of age a that has undergone s servicings must satisfy Bellman's equation

$$\begin{aligned} V(a, s) = \max\{ \quad & p(0, 0) - c + \delta V(1, 0), \\ & p(a, s+1) - k + \delta V(a+1, s+1), \\ & p(a, s) + \delta V(a+1, s)\}. \end{aligned}$$

Bellman's equation asserts that if the manufacturer keeps an asset of age a, he earns $p(a)$ over the coming year and begins the subsequent year with an asset worth $V(a+1)$; if he replaces the asset, on the other hand, he earns $p(0) - c$ over the coming year and begins the subsequent year with an asset worth $V(1)$. Actually, our language is a little loose here. The value $V(a)$ measures not only the current and future earnings of an asset of age a, but also the optimal earnings of all future assets that replace it.

7.2.4 Option Pricing

An American put option gives the holder the right, but not the obligation, to sell a specified quantity of a commodity at a specified strike price on or before a specified expiration date. In the Cox-Ross-Rubinstein binomial option pricing model, the price of the commodity is assumed to follow a two-state discrete jump process. Specifically, if the price of the commodity is p in period t, then its price in period $t+1$ will be pu with probability q and p/u with probability $1-q$ where:

$$\begin{aligned} u &= \exp(\sigma\sqrt{\Delta t}) > 1 \\ q &= \tfrac{1}{2} + \frac{\sqrt{\Delta t}}{2\sigma}\left(r - \tfrac{1}{2}\sigma^2\right) \\ \delta &= \exp(-r\Delta t). \end{aligned}$$

Here, r is the annualized interest rate, continuously compounded, σ is the annualized volatility of the commodity price, and Δt is the length of a period in years. Assuming the current price of the commodity is p_0, what is the value of an American put option if it has a strike price $\bar{p}$ and if it expires T years from today?

This is a finite horizon, stochastic model where time $t \in \{0, 1, 2, \ldots, N\}$ is measured in periods of length $\Delta t = T/N$ years each. The state is[2]

$$\begin{aligned} p &= \text{commodity price} \\ p &\in S = \{p_1 u^i | i = -N-1, -N, \ldots, N, N+1\}. \end{aligned}$$

The action is

$$\begin{aligned} x &= \text{decision to keep or exercise} \\ x &\in X = \{\text{keep}, \text{exercise}\}; \end{aligned}$$

the state transition probability rule is

$$P(p'|x,p) = \begin{cases} q & p' = pu \\ 1-q & p' = p/u \\ 0 & \text{otherwise} \end{cases}$$

the reward function is

$$f(p,x) = \begin{cases} 0 & x = \text{keep} \\ \bar{p} - p & x = \text{exercise} \end{cases}$$

[2]In this example, we alter our notation to conform with standard treatments of option valuation. Thus, the state is the price, denoted by p, the number of time periods until expiration is N, and T reserved for the time to expiration (in years).

The value function

$$V_t(p) = \text{option value at } t\text{, if commodity price is } p,$$

must satisfy Bellman's equation

$$V_t(p) = \max\{\ \bar{p} - p,\ q\delta V_{t+1}(pu) + (1-q)\delta V_{t+1}(p/u)\ \}$$

subject to the post-terminal condition

$$V_{N+1}(p) = 0$$

Note that if the option is exercised, the owner receives $\bar{p}-p$. If he does not exercise the option, however, he earns no immediate reward but will have an option in hand the following period worth $V_{t+1}(pu)$ with probability q and $V_{t+1}(p/u)$ with probability $1-q$. In option expires in the terminal period, making it valueless the following period; as such, the post-terminal salvage value is zero.

7.2.5 Job Search

At the beginning of each week, an infinitely-lived worker finds himself either employed or unemployed and must decide whether to be active in the labor market over the coming week by working, if he is employed, or by searching for a job, if he is unemployed. An active employed worker earns a wage w. An active unemployed worker earns an unemployment benefit u. An inactive worker earns a psychic benefit v from additional leisure, but no income. An unemployed worker that looks for a job will find one with probability p by the end of the week. An employed worker that remains at his job will be fired with probability q at the end of the week. What is the worker's optimal labor policy?

This is a infinite horizon, stochastic model with time $t = \{1, 2, \ldots, \infty\}$ measured in weeks. The state is

$$\begin{aligned} s &= \text{employment state} \\ s &\in S = \{\text{unemployed}(0), \text{employed}(1)\} \end{aligned}$$

and the action is

$$\begin{aligned} x &= \text{labor force participation decision} \\ x &\in X = \{\text{inactive}(0), \text{active}(1)\}. \end{aligned}$$

The state transition probability rule is

$$P(s'|s,x) = \begin{cases} 1 & x=0, s'=0 & \text{(inactive worker)} \\ 1-p & x=1, s=0, s'=0 & \text{(searches, finds no job)} \\ p & x=1, s=0, s'=1 & \text{(searches, finds job)} \\ q & x=1, s=1, s'=0 & \text{(works, loses job)} \\ 1-q & x=1, s=1, s'=1 & \text{(works, keeps job)} \\ 0 & \text{otherwise;} \end{cases}$$

and the reward function is

$$f(s,x) = \begin{cases} v & x=0 & \text{(inactive, receives leisure)} \\ u & x=1, s=0 & \text{(searching, receives benefit)} \\ w & x=1, s=1 & \text{(working, receives wage)} \end{cases}$$

The value function

$V(s)$ = Value of being in employment state s at beginning of week,

must satisfy Bellman's equation

$$V(s) = \begin{cases} \max\{v + \delta V(0), u + \delta p V(1) + \delta(1-p)V(0)\}, & s=0 \\ \max\{v + \delta V(0), w + \delta q V(0) + \delta(1-q)V(1)\}, & s=1 \end{cases}$$

7.2.6 Optimal Irrigation

Water from a dam can be used for either irrigation or recreation. Irrigation during the spring benefits farmers, but reduces the dam's water level during the summer, damaging recreational users. Specifically, farmer and recreational user benefits in year t are, respectively, $F(x_t)$ and $G(y_t)$, where x_t are the units of water used for irrigation and y_t are the units of water remaining for recreation. Water levels are replenished by random rainfall during the winter. With probability p, it rains one unit; with probability $1-p$ is does not rain at all. The dam has a capacity of M units of water and excess rainfall flows out of the dam without benefit to either farmer or recreational user. Derive the irrigation flow policy that maximizes the sum of farmer and recreational user benefits over an infinite time horizon.

This is a infinite horizon, stochastic model with time $t = \{1, 2, \ldots, \infty\}$ measured in years. The state is

$$\begin{aligned} s &= \text{units of water in dam at beginning of year} \\ s &\in S = \{0, 1, 2, \ldots, M\} \end{aligned}$$

and

$$\begin{aligned} x &= \text{units of water released for irrigation during year} \\ x &\in X(s) = \{0, 1, 2, \ldots, s\}. \end{aligned}$$

The state transition probability rule is

$$P(s'|s,x) = \begin{cases} p & s' = \min(s - x + 1, M) \quad \text{(rain)} \\ 1 - p & s' = s - x, \quad \text{(no rain)} \\ 0 & \text{otherwise} \end{cases}$$

and the reward function is

$$f(s,x) = F(x) + G(s - x).$$

The value function

$$V(s) = \text{Value of } s \text{ units of water in dam at beginning of year } t.$$

must satisfy Bellman's equation:

$$V(s) = \max_{x=0,1,\ldots,s} \{f(s,x) + \delta p V(\min(s - x + 1, M)) + \delta(1 - p)V(s - x)\}.$$

7.2.7 Optimal Growth

Consider an economy comprising a single composite good. Each year t begins with a predetermined amount of the good s_t, of which an amount x_t is invested and the remainder is consumed. The social welfare derived from consumption in year t is $u(s_t - x_t)$. The amount of good available in year $t + 1$ is $s_{t+1} = \gamma x_t + \epsilon_{t+1} f(x_t)$ where γ is the capital survival rate (1 minus the depreciation rate), f is the aggregate production function, and ϵ_{t+1} is a positive production shock with mean 1. What consumption-investment policy maximizes the sum of current and expected future welfare over an infinite horizon?

This is an infinite horizon, stochastic model with time $t \in \{0, 1, 2, \ldots\}$ measured in years. The model has a single state variable

$$\begin{aligned} s_t &= \text{stock of good at beginning of year } t \\ s_t &\in [0, \infty) \end{aligned}$$

and a single action variable

$$x_t = \text{amount of good invested in year } t$$

subject to the constraint

$$0 \leq x_t \leq s_t.$$

The reward earned by the optimizing agent is

$$u(s_t - x_t) = \text{social utility in } t.$$

State transitions are governed by

$$s_{t+1} = \gamma x_t + \epsilon_{t+1} f(x_t)$$

where

$$\epsilon_t = \text{productivity shock in year } t.$$

The value function, which gives the sum of current and expected future social welfare, satisfies Bellman's equation

$$V(s) = \max_{0 \leq x \leq s} \{u(s - x) + \delta E V(\gamma x + \epsilon f(x))\}, \qquad s > 0.$$

7.2.8 Renewable Resource Problem

A social planner wishes to maximize the discounted sum of net social surplus from harvesting a renewable resource over an infinite horizon. For year t, let s_t denote the resource stock at the beginning of the year, let x_t denote the amount of the resource harvested, let $c_t = c(x_t)$ denote the total cost of harvesting, and let $p_t = p(x_t)$ denote the market clearing price. Growth in the stock level is given by $s_{t+1} = g(s_t - x_t)$. What is the socially optimal harvest policy?

This is an infinite horizon, deterministic model with time $t \in \{0, 1, 2, \ldots\}$ measured in years. There is one state variable,

$$\begin{aligned} s_t &= \text{stock of resource at beginning of year } t \\ s_t &\in [0, \infty), \end{aligned}$$

and one action variable,

$$x_t = \text{amount of resource harvested in year } t,$$

subject to the constraint

$$0 \leq x_t \leq s_t.$$

The net social surplus is

$$\int_0^{x_t} p(\xi)\ d\xi - c(x_t).$$

State transitions are governed by

$$s_{t+1} = g(s_t - x_t).$$

The value function, which gives the net social value of resource stock, satisfies Bellman's equation

$$V(s) = \max_{0 \leq x \leq s} \{ \int_0^x p(\xi)\ d\xi - c(x) + \delta V(g(s-x)) \}.$$

7.2.9 Bioeconomic Model

In order to survive, an animal must forage for food in one of m distinct areas. In area x, the animal survives predation with probability p_x, finds food with probability q_x, and, if it finds food, gains e_x energy units. The animal expends one energy unit every period and has a maximum energy carrying capacity $\bar{s}$. If the animal's energy stock drops to zero, it dies. What foraging pattern maximizes the animal's probability of surviving T years to reproduce at the beginning of period $T+1$?

This is a finite horizon, stochastic model with time $t = \{1, 2, \ldots, T\}$ measured in foraging periods. The state is

$$\begin{aligned} s &= \text{stock of energy} \\ s &\in S = \{0, 1, 2, \ldots, \bar{s}\}; \end{aligned}$$

the action is

$$\begin{aligned} x &= \text{foraging area} \\ x &\in X = \{1, 2, \ldots, m\}. \end{aligned}$$

The state transition probability rule is, for $s = 0$,

$$P(s'|s,x) = \begin{cases} 1 & s' = 0 \qquad \text{(death is permanent)} \\ 0 & \text{otherwise;} \end{cases}$$

and, for $s > 0$,

$$P(s'|s,x) = \begin{cases} p_x q_x & s' = \min(\bar{s}, s - 1 + e_x) & \text{(survive, finds food)} \\ p_x(1-q_x) & s' = s-1 & \text{(survive, no food)} \\ (1-p_x) & s' = 0 & \text{(does not survive)} \\ 0 & \text{otherwise.} & \end{cases}$$

The reward function is

$$f(s,x) = 0.$$

Here, $s = 0$ is an absorbing state that, once entered, is never exited. More to the point, an animal whose energy stocks fall to zero dies, and remains dead. The reward function for periods 1 through T is zero, because there is only one payoff, surviving to procreate, and this payoff is earned in period $T + 1$.

The value function

$$V_t(s) = \text{probability of procreating, given energy stocks } s \text{ in period } t$$

must satisfy Bellman's equation

$$V_t(s) = \max_{x \in X} \{p_x q_x V_{t+1}(\min(\bar{s}, s - 1 + e)) + p_x(1 - q_x)V_{t+1}(s - 1)\},$$

for $t \in 1, \ldots, T$, with $V_t(0) = 0$, subject to the terminal condition

$$V_{T+1}(s) = \begin{cases} 0 & s = 0 \\ 1 & s > 0 \end{cases}$$

7.3 Solution Algorithms

Below, we develop numerical solution algorithms for stochastic discrete time, discrete space Markov decision models. The algorithms apply to deterministic models as well, provided one views a deterministic model as a degenerate special case of the stochastic model for which the transition probabilities are all zeros or ones.

To develop solution algorithms, we must introduce some vector notation and operations. Assume that the states $S = \{1, 2, \ldots, n\}$ and actions $X = \{1, 2, \ldots, m\}$ are indexed by the first n and m integers, respectively. Let $v \in \Re^n$ denote an arbitrary value vector:

$$v_i \in \Re = \text{value in state } i;$$

and let $x \in X^n$ denote an arbitrary policy vector:

$$x_i \in X = \text{action in state } i.$$

Also, for each policy $x \in X^n$, let $f(x) \in \Re^n$ denote the n-vector of rewards earned in each state when one follows the prescribed policy:

$$f_i(x) = \text{reward in state } i, \text{ given action } x_i \text{ taken};$$

and let $P(x) \in \Re^{n \times n}$ denote the n-by-n state transition probabilities when one follows the prescribed policy:

$$P_{ij}(x) = \text{probability of jump from state } i \text{ to } j, \text{ given action } x_i \text{ is taken}.$$

Given this notation, it is possible to express Bellman's equation for the finite horizon model succinctly as a recursive vector equation. Specifically, if $v_t \in \Re^n$ denotes the value function in period t, then

$$v_t = \max_x \{f(x) + \delta P(x) v_{t+1}\},$$

were the maximization is the vector operation induced by maximizing each row individually. Given the recursive nature of the finite horizon Bellman equation, one may compute the optimal value and policy functions v_t and x_t using backward recursion:

Algorithm: Backward Recursion

0. Initialization: Specify the rewards f, transition probabilities P, discount factor δ, terminal period T, and post-terminal value function v_{T+1}; set $t \leftarrow T$.

1. Recursion Step: Given v_{t+1}, compute v_t and x_t:

$$\begin{aligned} v_t &\leftarrow \max_x \{f(x) + \delta P(x) v_{t+1}\} \\ x_t &\leftarrow \operatorname*{argmax}_x \{f(x) + \delta P(x) v_{t+1}\}. \end{aligned}$$

2. Termination Check: If $t = 1$, stop; otherwise set $t \leftarrow t - 1$ and return to step 1.

Each recursive step involves a finite number of matrix-vector operations, implying that the finite horizon value functions are well-defined for every period. Note however, that it may be possible to have more than one sequence of optimal policies if ties occur in Bellman's equation. Since the algorithm requires exactly T iterations, it terminates in finite time with the value functions precisely computed and at least one optimal policy obtained.

Consider now the infinite horizon Markov decision model. Given the notation above, it is also possible to express the infinite horizon Bellman equation as a vector fixed-point equation

$$v = \max_x \{f(x) + \delta P(x) v\}.$$

This vector equation may be solved using standard function iteration methods:

Algorithm: Function Iteration

0. Initialization: Specify the rewards f, transition probabilities P, discount factor δ, convergence tolerance τ, and initial guess for the value function v.

1. Function Iteration: Update the value function v:

$$v \leftarrow \max_x \{f(x) + \delta P(x) v\}.$$

2. Termination Check: If $||\Delta v|| < \tau$, set

$$x \leftarrow \operatorname*{argmax}_x \{f(x) + \delta P(x) v\}$$

and stop; otherwise return to step 1.

Function iteration does not guarantee an exact solution in finitely many iterations. However, if the discount factor δ is less than one, the fixed-point map be shown to be a strong contraction. Thus, the infinite horizon value function exists and is unique, and may be computed to an arbitrary accuracy. Moreover, an explicit upper bound may be placed on the error associated with the final value function iterate. Specifically, if the algorithm terminates at iteration n, then

$$||v_n - v^*||_\infty \leq \frac{\delta}{1-\delta} ||v_n - v_{n-1}||_\infty$$

where v^* is the true value function.

The Bellman vector fixed-point equation for an infinite horizon model may alternatively be recast at a rootfinding problem

$$v - \max_x \{f(x) + \delta P(x) v\} = 0$$

and solved using Newton's method. By the Envelope Theorem, the derivative of the left-hand-side with respect to v is $I - \delta P(x)$ where x is optimal for the embedded maximization problem. As such, the Newton iteration rule is

$$v \leftarrow v - (I - \delta P(x))^{-1} (v - f(x) - \delta P(x) v)$$

where P and f are evaluated at the optimal x. After algebraic simplification the update rule may be written

$$v \leftarrow (I - \delta P(x))^{-1} f(x).$$

Newton's method applied to Bellman's equation traditionally has been referred to as 'policy iteration':

Algorithm: Policy Iteration

0. Initialization: Specify the rewards f, transition probabilities P, discount factor δ, and an initial guess for v.

1. Policy Iteration: Given the current value approximant v, update the policy x:

$$x \leftarrow \underset{x}{\operatorname{argmax}}\{f(x) + \delta P(x)v\}$$

   and then update the value by setting

$$v \leftarrow (I - \delta P(x))^{-1} f(x).$$

2. Termination Check: If $\Delta v = 0$, stop; otherwise return to step 1.

At each iteration, policy iteration either finds the optimal policy or offers a strict improvement in the value function. Because the total number of states and actions is finite, the total number of admissible policies is also finite, guaranteeing that policy iteration will terminate after finitely many iterations with an exact optimal solution. Policy iteration, however, requires the solution of a linear equation system. If $P(x)$ is large and dense, the linear equation could be expensive to solve, making policy iteration slow and possibly impracticable. In these instances, the function iteration algorithm may be the better choice.

The backward recursion, function iteration, and policy iteration algorithms are structured as a series of three nested loops. The outer loop represents either a backward recursion, function iteration, or policy iteration; the middle loop represents visits to each state; and the inner loop represents visits to each action. The computational effort needed to solve a discrete Markov decision model is roughly proportional to the product of the number of times each loop must be executed. More precisely, if n_s is the number of states and n_x is the number of actions, then $n_s \cdot n_x$ total actions need to be evaluated with each outer iteration.

The computational effort needed to solve a discrete Markov decision model is particularly sensitive to the dimensionality of the state and action variables. Suppose, for the sake of argument, that the state variable is k-dimensional and each dimension of the state variable has l different levels. Then the number of states will equal $n_s = l^k$. This implies that the computational effort required to solve the discrete Markov decision model will grow exponentially, not linearly, with the dimensionality of the state space. The same will be true regarding the dimensionality of the action space. The tendency for the solution time to grow exponentially with the dimensionality of the state or action space is called the "Curse of Dimensionality". Historically, the curse has represented the most severe practical problem encountered in solving discrete Markov decision models.

7.4 Dynamic Simulation Analysis

The optimal value and policy functions provide some insight into the nature of the controlled dynamic economic process. The optimal value function describes the benefits of being in a given state and the optimal policy function prescribes the optimal action to be taken there. However, the optimal value and policy functions provide only a partial, essentially static, picture of the controlled dynamic process. Typically, one wishes to analyze the controlled process further to learn about its dynamic behavior. Furthermore, one often wishes to know how the process is affected by changes in model parameters.

To analyze the dynamics of the controlled process, one will typically perform dynamic path and steady-state analysis. Dynamic path analysis examines how the controlled dynamic process evolves over time starting from some initial state. Specifically, dynamic path analysis describes the path or expected path followed by the state or some other endogenous variable and how the path or expected path will vary with changes in model parameters.

Steady-state analysis examines the longrun tendencies of the controlled process over an infinite horizon, without regard to the path followed over time. Steady-state analysis of a deterministic model seeks to find the values to which the state or other endogenous variables will converge over time, and how the limiting values will vary with changes in the model parameters. Steady-state analysis of a stochastic model requires derivation of the steady-state distribution of the state or other endogenous variable. In many cases, one is satisfied to find the steady-state means and variances of these variables and their sensitivity to changes in exogenous model parameters.

The path followed by a controlled, finite horizon, deterministic, discrete, Markov decision process is easily computed. Given the state transition function g and the optimal policy functions x_t^*, the path taken by the state from an initial point s_1 can be computed as follows:

$$\begin{aligned}
s_2 &= g(s_1, x_1^*(s_1)) \\
s_3 &= g(s_2, x_2^*(s_2)) \\
s_4 &= g(s_3, x_3^*(s_3)) \\
&\vdots \\
s_{T+1} &= g(s_T, x_T^*(s_T)).
\end{aligned}$$

Given the path of the controlled state, it is straightforward to derive the path of actions through the relationship $x_t = x_t^*(s_t)$. Similarly, given the path taken by the controlled state and action allows one to derive the path taken by any function of the state and action.

A controlled, infinite horizon, deterministic, discrete Markov decision process can be analyzed similarly. Given the state transition function g and optimal policy function x^*, the path taken by the controlled state from an initial point s_1 can be computed from the iteration rule:

$$s_{t+1} = g(s_t, x^*(s_t)).$$

The steady-state of the controlled process can be computed by continuing to form iterates until they converge. The path and steady-state values of other endogenous variables, including the action variable, can then be computed from the path and steady-state of the controlled state.

Analysis of controlled, stochastic, discrete Markov decision processes is a bit more complicated because such processes follow a random, not a deterministic, path. Consider a finite horizon process whose optimal policy x_t^* has been derived for each period t. Under the optimal policy, the controlled state will be a finite horizon Markov chain with nonstationary transition probability matrices P_t^*, whose row i, column j element is the probability of jumping from state i in period t to state j in period $t+1$, given that the optimal policy $x_t^*(i)$ is followed in period t:

$$P_{tij}^* = \Pr(s_{t+1} = j | x_t = x_t^*(i), s_t = i)$$

The controlled state of an infinite horizon, stochastic, discrete Markov decision model with optimal policy x^* will be an infinite horizon stationary Markov chain with transition probability matrix P^* whose row i, column j element is the probability of jumping from state i in one period t to state j in the following period, given that the optimal policy $x^*(i)$ is followed:

$$P_{ij}^* = \Pr(s_{t+1} = j | x_t = x^*(i), s_t = i)$$

Given the transition probability matrix P^* for the controlled state it is possible to simulate a representative state path, or, for that matter, many representative state paths, by performing Monte Carlo simulation. To perform Monte Carlo simulation, one picks an initial state, say s_1. Having the simulated state $s_t = i$, one may simulate a jump to s_{t+1} by randomly picking a new state j with probability P_{ij}^*.

The path taken by the controlled state of an infinite horizon, stochastic, discrete Markov model may also be described probabilistically. To this end, let Q_t denote the matrix whose row i, column j entry gives the probability that the process will be in state j in period t, given that it is in state i in period 0. Then the t-period transition probability matrices Q_t are simply the matrix powers of P:

$$Q_t = P^t$$

where $Q_0 = I$. Given the t-period transition probability matrices Q_t, one can fully describe, in a probabilistic sense, the path taken by the controlled process from any initial state $s_0 = i$ by looking at the i^{th} rows of the matrices Q_t.

In most economic applications, the multiperiod transition matrices Q_t will converge to a matrix Q as t goes to infinity. In such cases, each entry of Q will indicate the relative frequency with which the controlled decision process will visit a given state in the longrun, when starting from given initial state. In the event that all the columns of Q are identical and the longrun probability of visiting a given state is independent of initial state, then we say that the controlled state process possesses a steady-state distribution. The steady state distribution is given by the probability vector π that is the common row of the matrix Q. Given the steady-state distribution of the controlled state process, it becomes possible to compute summary measures about the longrun behavior of the controlled process, such as its longrun mean or variance. Also, it is possible to derive the longrun probability distribution of the optimal action variable or the longrun distribution of any other variables that are functions of the state and action.

7.5 A Discrete Dynamic Programming Toolbox

In order to simplify the process of solving discrete Markov decision models, we have provided a single, unifying routine `ddpsolve` that solves such models using the dynamic programming algorithm selected by the user. The routine is executed by issuing the following command:

```
[v,x,pstar] = ddpsolve(model,alg,v)
```

Here, on input, `model` is a structured variable that contains all relevant model information, including the time horizon, the discount factor, the reward matrix, the probability transition matrix, and the terminal value function (if needed); `alg` is a string that specifies the algorithm to be used, either 'newt' for policy iteration, 'func' for function iteration, or 'back' for backward recursion; and `v` is the post-terminal value function, if the model has finite horizon, or an initial guess for the value function, if the model has infinite horizon. On output, `v` is the optimal value function, `x` is the optimal policy, and `pstar` is the optimal probability transition matrix.

The structured variable `model` contains five fields, `horizon`, `discount`, `reward`, `transition`, and `vterm` which are specified as follows:

- `horizon -` The time horizon, a positive integer or 'inf'.
- `discount -` The discount factor, a positive scalar less than one.
- `reward -` An n by m matrix of rewards whose rows and columns are associated with states and actions, respectively.

- `transition` - An mn by n matrix of state transition probabilities whose rows are associated with this period's state and whose columns are associated with next period's state. The state transition probability matrices for the various actions are stacked vertically on top of each other, with the n by n transition probability matrix associated with action 1 at the top and the n by n transition probability matrix associated with action m at the bottom.
- `vterm` - An n by 1 vector of terminal values, if model has a finite horizon, or initial guess for value function, if model has an infinite horizon. It has a default value of zero if not specified.

The routine `ddpsolve` implements all three standard solution algorithms relying on two elementary routines. One routine takes the current value function `v`, the reward matrix `f`, the probability transition matrix `P`, and the discount factor `delta` and solves the optimization problem embedded in Bellman's equation, yielding an updated value function `v` and optimal policy `x`:

```
function [v,x] = valmax(v,f,P,delta)
[m,n]=size(f);
[v,x]=max(f+delta*reshape(P*v,m,n),[],2);
```

The second routine takes a policy `x`, the reward matrix `f`, the probability transition matrix `P`, and the discount factor `delta` and returns the state reward function `fstar` and state probability transition matrix `Pstar` induced by the policy:

```
function [pstar,fstar] = valpol(x,f,P,delta)
[n,m]=size(f); i=(1:n)';
pstar = P(n*(x(i)-1)+i,:);
fstar = f(n*(x(i)-1)+i);
```

Given the `valmax` and `valpol` routines, it is straightforward to implement the backward recursion, function iteration, and policy iteration algorithms used to solve discrete Markov decision models. The MATLABscript that performs backward recursion for a finite horizon model is

```
[n,m]=size(f);
x = zeros(n,T);
v = [zeros(n,T) vterm];
for t=T:-1:1
   [v(:,t),x(:,t)] = valmax(v(:,t+1),f,P,delta);
end
```

The MATLABscript that performs function iteration for the infinite horizon model is

```
for it=1:maxit
   vold = v;
   [v,x] = valmax(v,f,P,delta);
   if norm(v-vold)<tol, return, end;
end
```

The MATLABscript that performs policy iteration for the infinite horizon model is

```
for it=1:maxit
   vold = v;
   [v,x] = valmax(v,f,P,delta);
   [pstar,fstar] = valpol(x,f,P,delta);
   v = (eye(n,n)-delta*pstar)\fstar;
   if norm(v-vold)<tol, return, end;
end
```

The toolbox accompanying the textbook also provides two utilities for performing dynamic analysis. The first routine, `ddpsimul` is employed as follows:

```
st = ddpsimul(pstar,s1,nyrs,x)
```

On input, `pstar` is the optimal probability transition matrix induced by the optimal policy, which is generated by the routine `ddpsolve`; `x` is the optimal policy, which is also generated by the routine `ddpsolve`; `s1` is a k by 1 vector of initial states, each entry of which initiates a distinct replication of the optimized state process; and `nyrs` is the number of years for which the process will be simulated. On output, `st` is a `k` by `nyrs` vector containing `k` replications of the process, each `nyrs` in length. When the model is deterministic, the path is deterministic. When the model is stochastic, the path is generated by Monte Carlo methods. If we simulate replications all which begin from the same state, the row average of the vector `st` will provide an estimate of the expected path of the state.

The toolbox accompanying the textbook provides a second utility for performing dynamic analysis called `markov`, which is employed as follows:

```
pi=markov(pstar);
```

On input, `pstar` is the optimal probability transition matrix induced by the optimal policy, which is generated by the routine `ddpsolve`. On output, `pi` is a vector containing the invariant distribution of the optimized state process.

Finally, the toolbox accompanying the textbook provides a utility for converting the deterministic state transition rule into the equivalent degenerate probability transition matrix. The routine is employed as follows:

```
P = expandg(g);
```

On input, g is the deterministic state transition rule. On output, P is the corresponding probability transition matrix.

Given the aforementioned MATLAButilities, the most significant practical difficultly typically encountered when solving discrete Markov decision models is correctly initializing the reward and state transition matrices. We demonstrate how to implement these routines in practice in the following section.

7.6 Numerical Examples

7.6.1 Mine Management

Consider the mine management model with market price $p = 1$, initial stock of ore $\bar{s} = 100$, and annual discount factor $\delta = 0.95$.

The first step required to solve the model numerically is to specify the model parameters and to construct the state and action spaces:

```
delta = 0.9;                 % discount factor
price = 1;                   % price of ore
sbar = 100;                  % initial ore stock
S = (0:sbar)';               % vector of states
n = length(S);               % number of states
X = (0:sbar)';               % vector of actions
m = length(X);               % number of actions
```

Next, one constructs the reward and transition probability matrices:

```
f = zeros(n,m);
for k=1:m
   f(:,k) = price*X(k)-(X(k)^2)./(1+S);
   f(X(k)>S,k) = -inf;
end
g = zeros(n,m);
for k=1:m
  j = max(0,S-X(k)) + 1;
  g(:,k) = j;
end
P = expandg(g);
```

Notice that a reward matrix element is set to negative infinity if the extraction level exceeds the available stock. This guarantees that the value maximization algorithm will not chose an infeasible action. Also note that we have defined the deterministic state transition rule `g` first, and then used the utility `expandg` to construct the associated probability transition matrix, which consists of mostly zeros and is stored in sparse matrix format to accelerate subsequent computations.

One then packs the essential data into the structured variable `model`:

```
model.reward     = f;
model.transition = P;
model.horizon    = inf;
model.discount   = delta;
```

Once the model data have been specified, solution of the model is relatively straightforward. To solve the infinite horizon model via policy iteration, one issues the command:

```
[vi,xi,pstari] = ddpsolve(model);
```

To solve the infinite horizon model via function iteration, one issues the command:

```
[vi,xi,pstari] = ddpsolve(model,'func');
```

Upon convergence, `vi` will be `n` vector containing the value function and `xi` will be `n` vector containing the indices of the optimal ore extractions. Note that the policy iteration algorithm was not explicitly specified because it is the default algorithm when the horizon is infinite.

To solve the model over a ten year horizon, one issues the commands

```
model.horizon = 10;
[vf,xf,pstarf] = ddpsolve(model);
```

Note that we do not have to pass the post-terminal value function, since it is identically zero, the default. Also note that the backward recursion algorithm was not explicitly specified because it is the default algorithm when the horizon is finite. Upon completion, `xf` is an `n` by `10` matrix containing the optimal ore extraction policy for all possible ore stock levels for periods 1 to `10`. The columns of `x` represent periods and its rows represent states. Similarly, `vf` is an `n` by `11` matrix containing the optimal values for all possible stock levels for periods 1 to `11`.

Once the optimal solution has been computed, one may plot the optimal value and extraction policy functions:

```
figure(1); plot(S,X(xi));
xlabel('Stock'); ylabel('Optimal Extraction');
figure(2); plot(S,vi);
xlabel('Stock'); ylabel('Optimal Value');
```

Both functions are illustrated in Figure 7.1.

To analyze the dynamics of the optimal solution, one may also plot the optimal path of the stock level over time, starting from the initial stock level, for both the finite and infinite horizon models:

```
s1 = length(S); nyrs = 10;
sipath = ddpsimul(pstari,s1,nyrs,xi);
sfpath = ddpsimul(pstarf,s1,nyrs,xf);
figure(3)
plot(1:nyrs,S(sipath),1:nyrs,S(sfpath));
legend('Infinite Horizon','Ten Year Horizon');
xlabel('Year'); ylabel('Stock');
```

As seen in Figure 7.1, one extracts the stock at a faster rate if the horizon is finite.

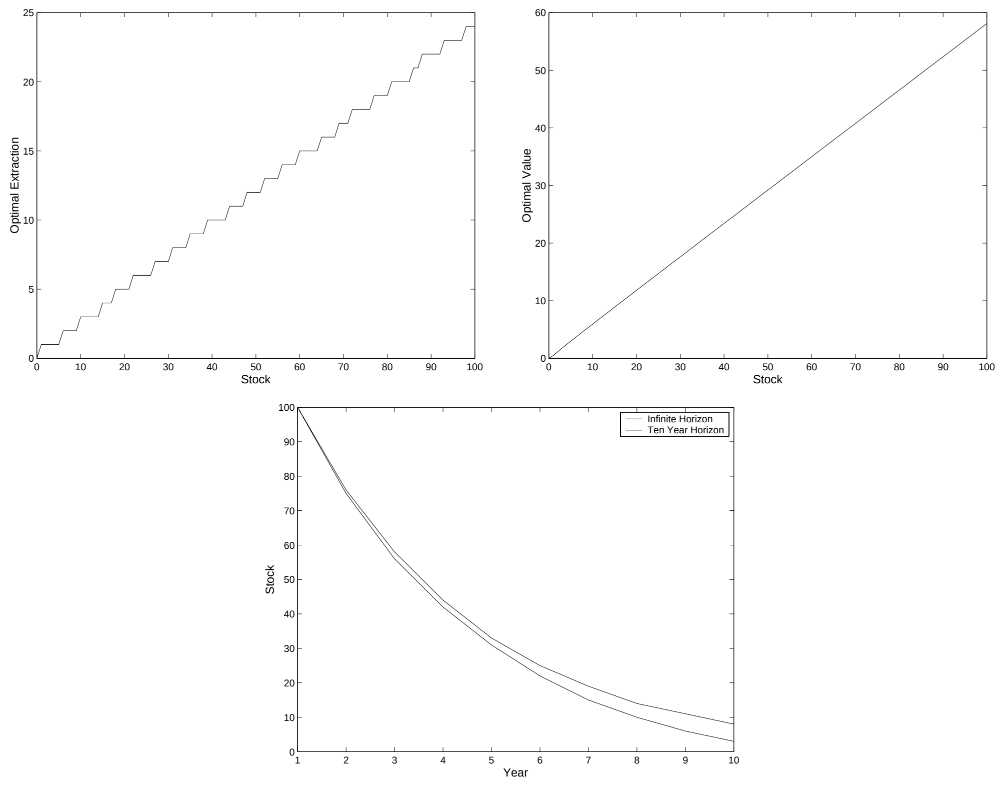

Figure 7.1: Solution to Mine Management Problem

7.6.2 Asset Replacement - I

Suppose that a new machine costs \$50 and that the net profit contribution of a machine is:

age	net profit
0	50
1	45
2	35
3	20
4+	0

Then, letting 0=keep and 1=replace, the optimal replacement policy over a five year planning horizon, with no discounting, is:

	Optimal Policy					Optimal Value				
	Machine Age					Machine Age				
Year	0	1	2	3	4+	0	1	2	3	4+
5	0	0	0	0	0	50.0	45.0	35.0	20.0	0.0
4	0	0	0	0	0	95.0	80.0	55.0	20.0	0.0
3	0	0	1	1	1	130.0	100.0	70.0	55.0	35.0
2	0	1	1	1	1	150.0	115.0	105.0	90.0	70.0
1	0	0	1	1	1	165.0	150.0	125.0	110.0	90.0

Assuming a discount factor of 0.9, the initial year optimal policy and value for functions for differing horizons are:

	Optimal Policy					Optimal Value				
	Machine Age					Machine Age				
Horizon	0	1	2	3	4+	0	1	2	3	4+
1	0	0	0	0	0	50.0	45.0	35.0	20.0	0.0
2	0	0	0	0	0	90.5	76.5	53.0	20.0	0.0
3	0	0	1	1	1	118.8	92.7	56.4	41.4	21.4
4	0	0	1	1	1	133.4	95.8	82.0	67.0	47.0
5	0	0	0	1	1	136.2	118.8	95.3	80.1	60.1
6	0	0	0	1	1	156.9	130.7	107.1	82.6	62.6
7	0	0	1	1	1	167.7	141.4	116.2	101.2	81.2
8	0	0	0	1	1	177.2	149.6	126.1	110.9	90.9
9	0	0	0	1	1	184.6	158.5	134.8	119.5	99.5
10	0	0	0	1	1	192.6	166.3	142.6	126.2	106.2
20	0	0	0	1	1	237.2	210.3	186.5	171.0	151.0
40	0	0	0	1	1	257.9	231.3	207.4	191.8	171.8

60	0	0	0	1	1	260.5	233.9	209.9	194.4	174.4
100	0	0	0	1	1	260.8	234.2	210.2	194.7	174.7
200	0	0	0	1	1	260.8	234.2	210.2	194.7	174.7

The optimal steady-state policy is to replace the tractor after year three.

7.6.3 Asset Replacement - II

Consider the same model as above, except that now the profit contribution of a tractor

$$f(a, n) = (50.0 - 2.5a - 2.5a^2) * (1 - (a - n)/4)$$

depends both on its age a and the number of times n it has undergone end-of-year servicing. At the beginning of the year, a farmer must decide what to do at the end of the year: keep and service the tractor, keep but not service the tractor, or replace the tractor. It costs \$75 to order a new tractor and \$10 to schedule one for servicing. Assuming a discount factor of 0.9, the steady-state optimal replacement-maintenance policy and value functions are:

	Optimal Policy					Optimal Value				
	Times Serviced					Times Serviced				
Age	0	1	2	3	4	0	1	2	3	4
0	2	0	0	0	0	163.2	0.0	0.0	0.0	0.0
1	1	2	0	0	0	114.2	136.8	0.0	0.0	0.0
2	3	1	1	0	0	89.3	99.9	113.2	0.0	0.0
3	3	3	3	3	0	76.8	81.8	86.8	91.8	0.0
4	3	3	3	3	3	71.8	71.8	71.8	71.8	71.8

where 0=not defined, 1=keep but don't service, 2=keep and service, and 3=replace.

The optimal steady-state policy is thus to service the tractor after its first and second years, to keep it but not service it after its third year, and to replace it after its fourth year. Should one forget to service the tractor after its first year, then it would be optimal to keep but not service it after its second year and then to sell it after its third year.

7.6.4 Option Pricing

Consider the binomial option pricing model with current asset price $p_1 = 2.00$, strike price $\bar{p} = 2.10$, annual interest rate $r = 0.05$, annual volatility $\sigma = 0.2$, and time to expiration $T = 0.5$ years that is to be divided into $N = 50$ intervals.

The first step required to solve the model numerically is to specify the model parameters and to construct the state space:

```
T = 0.5;                        % years to expiration
sigma = 0.2;                    % annual volatility
r = 0.05;                       % annual interest rate
strike = 2.1;                   % option strike price
p1 = 2;                         % current asset price
N = 100;                        % number of time intervals
tau = T/N;                      % length of time intervals
delta = exp(-r*tau);            % discount factor
u = exp( sigma*sqrt(tau));      % up jump factor
q = 0.5+tau^2*(r-(sigma^2)/2)/(2*sigma); % up jump probability
price = p1*(u.^(-N:N))';        % asset prices
n = length(price);              % number of states
```

There is no need to explicitly define an action space since actions are represented by integer indices.

Next, one constructs the reward and transition probability matrices:

```
f = [ strike-price zeros(n,1) ];
P = zeros(n,n);
for i=1:n
  P(i,min(i+1,n)) = q;
  P(i,max(i-1,1)) = 1-q;
end
P = [zeros(n,n); P];
P = sparse(P);
```

Here, action 1 is identified with the exercise decision and action 2 is identified with the hold decision. Note how the transition probability matrix associated with the decision to exercise the option is identically the zero matrix. This is done to ensure that the expected future value of an exercised option always computes to zero. Also note that because the probability transition matrix contains mostly zeros, it is stored in sparse matrix format to speed up subsequent computations.

One then packs the essential model data into a structured variable `model`:

```
model.reward     = f;
model.transition = P;
model.discount   = delta;
model.horizon    = N+1;
```

To solve the finite horizon model via backward recursion, one issues the command:

```
[v,x] = ddpsolve(model);
```

Upon completion, `v(:,1)` is an `n` vector that contains the value of the American option in period 1 for different asset prices.

Once the optimal solution has been computed, one may plot the optimal value function.

```
plot(price,v(:,1)); axis([0 strike*2 -inf inf]);
xlabel('Asset Price'); ylabel('Put Option Premium');
```

This plot is given in Figure 7.2.

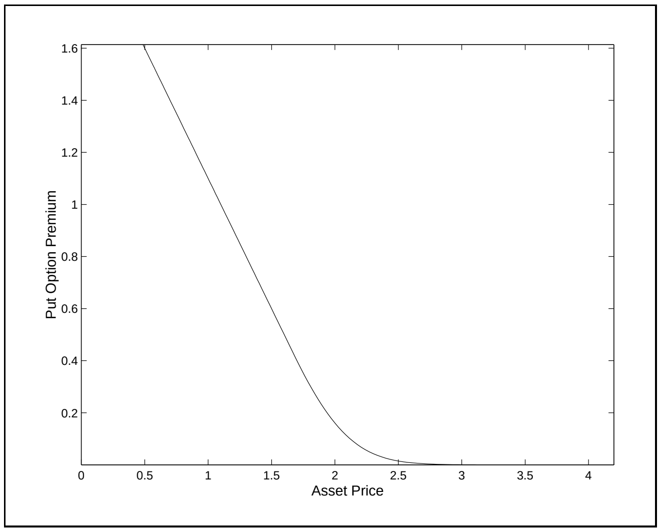

Figure 7.2

7.6.5 Job Search

Consider the job search model with weekly unemployment benefit $u = 55$ and psychic benefit from leisure $v = 60$. Also assume the probability of finding a job is $p = 0.90$, the probability of being fired is $q = 0.05$, and the weekly discount rate is $\delta = 0.99$. Suppose we wish to explore the optimal labor market participation policy for wages ranging from $w = 55$ to $w = 65$.

The first step required to solve the model numerically is to specify the model parameters:

```
u =  50;                          % weekly unemp. benefit
v =  60;                          % weekly value of leisure
pfind = 0.90;                     % prob. of finding job
```

```
pfire = 0.10;                  % prob. of being fired
delta = 0.99;                  % discount factor
```

Note that by identifying both states and actions with their integer indices, one does not need to explicitly generate the state and action space.

Next, one constructs the reward and transition probability matrices. Here, we identify state 1 with unemployment and state 2 with employment, and identify action 1 with inactivity and action 2 with participation:

```
f = zeros(2,2);
f(:,1) = v;                    % gets leisure
f(1,2) = u;                    % gets benefit
P1 = sparse(zeros(2,2));
P2 = sparse(zeros(2,2));
P1(:,1) = 1;                   % remains unemployed
P2(1,1) = 1-pfind;             % finds no job
P2(1,2) = pfind;               % finds job
P2(2,1) = pfire;               % gets fired
P2(2,2) = 1-pfire;             % keeps job
P = [P1;P2];
```

One then packs the essential model data into a structured variable `model`:

```
model.reward     = f;
model.transition = P;
model.horizon    = inf;
model.discount   = delta;
```

To solve the infinite horizon model via policy iteration at different wage rates, one issues the command :

```
xtable = [];
wage=55:65;
for w=wage
   f(2,2) = w; model.reward = f; % vary wage
   [v,x] = ddpsolve(model);      % solve via policy iteration
   xtable = [xtable x];          % tabulate
end
```

Upon convergence, `xtable` will be a matrix containing the optimal labor force participation decisions at different wage rates. The table may be printed by issuing the following commands:

```
fprintf('\nOptimal Job Search Strategy')
fprintf('\n  (1=inactive, 2=active)\n')
fprintf('\nWage  Unemployed  Employed\n')
fprintf('%4i  %10i%10i\n',[wage;xtable])
```

The optimal decision rule is given in Table 7.1.

Table 7.1: Optimal Labor Participation Rule

Wage	Unemployed	Employed
55	I	I
56	I	I
57	I	I
58	I	I
59	I	I
60	I	I
61	I	A
62	A	A
63	A	A
64	A	A
65	A	A

7.6.6 Optimal Irrigation

The first step required to solve the model numerically is to specify the model parameters and to construct the state and action spaces:

```
delta =  0.9;
irrben = [-3;5;9;11];          % Irrigation Benefits to Farmers
recben = [-3;3;5;7];           % Recreational Benefits to Users
maxcap = 3;                    % maximum dam capacity
S = (0:1:maxcap)';             % vector of states
n = length(S);                 % number of states
X = (0:1:maxcap)';             % vector of actions
m = length(X);                 % number of actions
```

Next, one constructs the reward matrix:

```
f = zeros(n,m);
```

```
for i=1:n;
for k=1:m;
   if k>i
     f(i,k) = -inf;
   else
     f(i,k) = irrben(k) + recben(i-k+1);
   end
end
end
```

Here, a reward matrix element is set to negative infinity if the irrigation level exceeds the available water stock, an infeasible action.

Next, one constructs the transition probability matrix:

```
P = [];
for k=1:m
   Pk = sparse(zeros(n,n));
   for i=1:n;
      j=i-k+1; j=max(1,j); j=min(n,j);
      Pk(i,j) = Pk(i,j) + 0.4;
      j=j+1; j=max(1,j); j=min(n,j);
      Pk(i,j) = Pk(i,j) + 0.6;
   end
   P = [P;Pk];
end
```

One then packs the essential model data into a structured variable `model`:

```
model.reward     = f;
model.transition = P;
model.horizon    = inf;
model.discount   = delta;
```

To solve the infinite horizon model via policy iteration, one issues the command:

```
[v,x] = ddpsolve(model);
```

To solve the infinite horizon model via function iteration, one issues the command:

```
[v,x] = ddpsolve(model,'func');
```

Upon convergence, `v` will be `n` vector containing the value function and `x` will be `n` vector containing the optimal irrigation policy.

Once the optimal solution has been computed, one may plot the optimal value and irrigation policy functions:

```
figure(1); plot(S,X(x));
xlabel('Stock'); ylabel('Optimal Irrigation');
figure(2); plot(S,v);
xlabel('Stock'); ylabel('Optimal Value');
```

Suppose one wished to compute the steady-state stock level. One could easily do this by calling `markov` to compute the steady state distribution and integrating:

```
pi = markov(pstar);
avgstock = pi'*S;
fprintf('\nSteady-state Stock        %8.2f\n',avgstock)
```

To plot expected water level over time given that water level is currently zero, one would issue the commands

```
figure(3)
nyrs = 20;
s1=ones(10000,1);
st = ddpsimul(pstar,s1,nyrs,x);
plot(1:nyrs,mean(S(st)));
xlabel('Year'); ylabel('Expected Water Level');
```

Here, we use the function `ddpsimul` to simulate the evolution of the water level via Monte Carlo 10000 times over a 20 year horizon. The mean of the 10000 replications is then computed and plotted for each year in the simulation. The expected path, together with the optimal value and policy functions are given in Figure 7.3.

7.6.7 Optimal Growth

7.6.8 Renewable Resource Problem

7.6.9 Bioeconomic Model

Consider the bioeconomic model with three foraging areas, predation survival probabilities $p_1 = 1$, $p_2 = 0.98$, and $p_3 = 0.90$, and foraging success probabilities $q_1 = 0$, $q_2 = 0.3$, and $q_3 = 0.8$. Also assume that successful foraging delivers $e = 4$ units of energy in all areas and that the procreation horizon is 10 periods.

The first step required to solve the model numerically is to specify the model parameters and to construct the state and action spaces:

```
T = 10;                          % foraging periods
eadd =  4;                       % energy from foraging
```

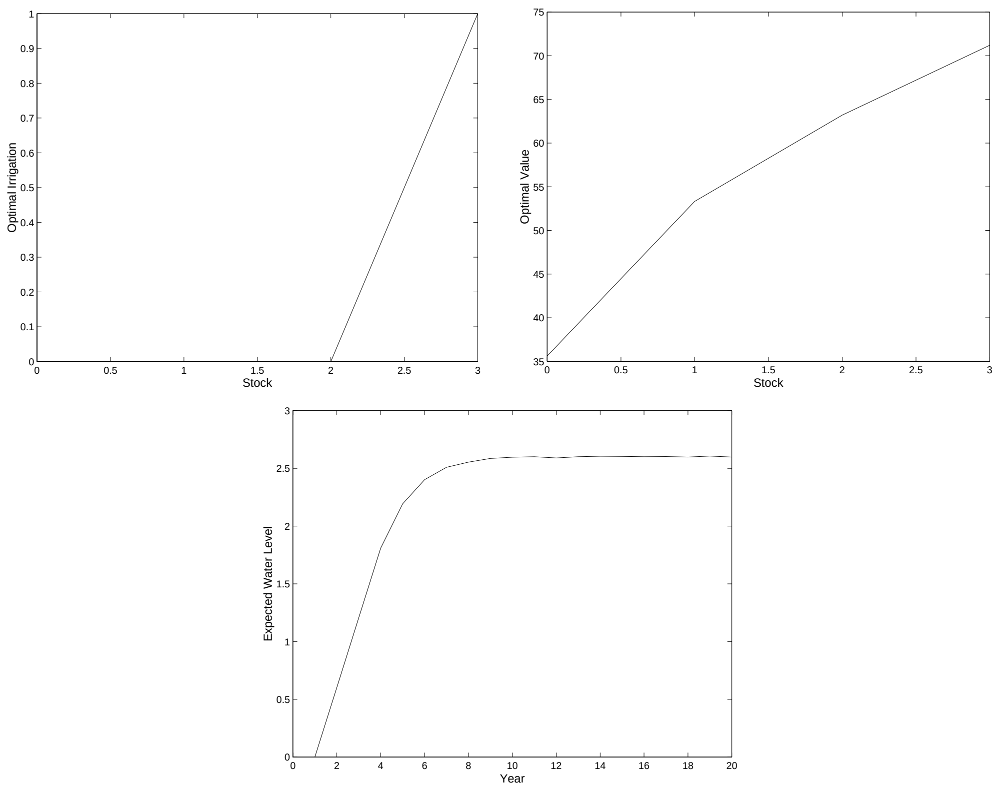

Figure 7.3: Solution to Optimal Irrigation Problem

```
emax = 10;                     % energy capacity
S = 0:emax;                    % energy levels
n = length(S);                 % number of states
X = 1:3;                       % foraging areas
m = length(X);                 % number of actions
```

There is no need to explicitly define an action space since actions are represented by integer indices.

Next, one constructs the reward and transition probability matrices:

```
f = zeros(n,m);
p = [1 .98 .9];                % predation survival prob.
q = [0 .30 .8];                % foraging success prob.
P = [];
for k=1:m
  Pk = zeros(n,n);
```

```
    Pk(1,1) = 1;
    for i=2:n;
       Pk(i,min(n,i-1+eadd)) = p(k)*q(k);
       Pk(i,i-1)             = p(k)*(1-q(k));
       Pk(i,1)               = Pk(i,1) + (1-p(k));
    end
    P = [ P ; Pk ];
  end
```

Note that the reward matrix is zero because the reward is not earned until the post-terminal period. Upon the reaching the post-terminal period, either the animal is alive, earning reward of 1, or is dead, earning a reward of 0. We capture this by specifying the terminal value function as follows

```
v = ones(n,1);                  % terminal value: survive
v(1) = 0;                       % terminal value: death
```

One then packs the essential model data into a structured variable `model`:

```
model.reward     = f;
model.transition = P;
model.horizon    = inf;
model.discount   = delta;
model.vterm      = v;
```

To solve the finite horizon model via backward recursion, one issues the command:

```
[v,x] = ddpsolve(model);
```

Upon convergence, `v` will be `n` by `1` matrix containing the value function and `ix` will be `n` by `1` matrix containing the indices of the optimal foraging policy for all possible initial energy stock levels.

Once the optimal solution has been computed, one may print out the survival probabilities (see Table 7.2):

```
fprintf('\nProbability of Survival\n')
disp('                       Stock of Energy')
fprintf('Period ');fprintf('%5i ',S);fprintf('\n');
for t=1:T
   fprintf('%5i ',t);fprintf('%6.2f',v(:,t)');fprintf('\n')
end
```

A similar script can be executed to print out the optimal foraging strategy (see Table 7.3).

Table 7.2: Survival Probabilities

	Stock of Energy										
Period	0	1	2	3	4	5	6	7	8	9	10
1	0.00	0.59	0.71	0.80	0.82	0.83	0.85	0.92	0.93	0.93	0.93
2	0.00	0.59	0.77	0.80	0.82	0.83	0.92	0.92	0.93	0.93	1.00
3	0.00	0.64	0.77	0.80	0.82	0.91	0.92	0.92	0.93	1.00	1.00
4	0.00	0.64	0.77	0.80	0.90	0.91	0.92	0.92	1.00	1.00	1.00
5	0.00	0.64	0.77	0.88	0.90	0.91	0.92	1.00	1.00	1.00	1.00
6	0.00	0.64	0.85	0.88	0.90	0.91	1.00	1.00	1.00	1.00	1.00
7	0.00	0.72	0.85	0.88	0.90	1.00	1.00	1.00	1.00	1.00	1.00
8	0.00	0.72	0.85	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00
9	0.00	0.72	0.85	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
10	0.00	0.72	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 7.3: Optimal Foraging Strategy

	Stock of Energy										
Period	0	1	2	3	4	5	6	7	8	9	10
1	1	3	3	3	2	2	2	2	2	2	2
2	1	3	3	3	2	2	2	2	2	2	1
3	1	3	3	3	2	2	2	2	2	1	1
4	1	3	3	3	2	2	2	2	1	1	1
5	1	3	3	2	2	2	2	1	1	1	1
6	1	3	3	2	2	2	1	1	1	1	1
7	1	3	3	2	2	1	1	1	1	1	1
8	1	3	3	2	1	1	1	1	1	1	1
9	1	3	3	1	1	1	1	1	1	1	1
10	1	3	1	1	1	1	1	1	1	1	1

Exercises

7.1. Consider a stationary 3-state Markov chain with transition probability matrix:

$$P = \begin{bmatrix} 0.2 & 0.4 & 0.4 \\ 0.5 & 0.5 & 0.0 \\ 0.6 & 0.2 & 0.2 \end{bmatrix}.$$

(a) Is the Markov chain irreducible?

(b) If so, find the steady-state distribution.

7.2. A machine lasts a maximum of 6 years but may require replacement sooner. Suppose that the probability of requiring replacement after each year is given by

Cycle	Prob
1	0.03
2	0.04
3	0.12
4	0.39
5	0.80
6	1.00

(a) What is the age distribution of macines in a large population? Draw a histogram.

(b) What is the average machine age in a large population?

7.3. A firm operates in an uncertain profit environment. The firm takes an operating loss of one unit in a bad year, it makes a operating profit of two units in an average year, and it makes an operating profit of four units in a good year. At the beginning of a bad year, the firm may elect to shut down, avoiding the operating loss. Although the firm faces no fixed costs or shut-down costs, it incurs a start-up cost 0.2 units if it reopens after one or more periods of inactivity. The profit environment follows a stationary first-order Markov process with transition probabilities:

```
        to
bad  avg  good
```

	bad	0.4	0.5	0.1
from	avg	0.3	0.4	0.3
	good	0.1	0.5	0.4

(a) Suppose the firm adopts the policy of staying open regardless of the profit environment in any given year. Given that this is a bad year, how much profit can the firm expect to make one year from now, two years from now, three years from now, ten years from now?

(b) Suppose the firm adopts the following policy: (i) in a bad year, do not operate; (ii) in a good year, operate; and (iii) in an average year, do what you did the preceding year. Given that this is a bad year, how much profit can the firm expect to make one year from now, two years from now, three years from now?

Graph the expected profits for both parts on the same figure.

7.4. Consider a competitive price-taking firm that wishes to maximize the present value sum of current and future profits from harvesting a nonrenewable resource. In year t, the firm earns revenue $p_t x_t$ where p_t is the market price for the harvested resource and x_t is the amount harvested by the firm; the firm also incurs cost αx_t^β, where α and β are cost function parameters. The market price takes one of two values, p_1 or p_2, according to the first-order Markov probability law:

$$\Pr[p_{t+1} = p_j | p_t = p_i] = w_{ij}.$$

Assuming an annual discount factor of δ, and that harvest levels and stocks must be integers, formulate the firm's optimization problem. Specifically, formulate Bellman's functional equation, clearly identifying the state and action variables, the state and action spaces, and the reward and probability transition functions.

7.5. Consider a timber stand that grows by one unit of biomass per year. That is, if the stand is planted with seedlings at the beginning of year t, it will contain $t' - t$ units of biomass in year t'. Harvesting decisions are made at the beginning of each year. If the stand is harvested, new seedlings are replanted at the end of the period (so the stand has biomass 0 in the next period). The price of harvested timber is p dollars per unit and the cost of harvesting and replanting is c. The timber firm discounts the future using a discount factor of δ.

(a) Set up the decision problem (define states, controls, reward function, transition rule).

(b) Formulate the value function and Bellman's recursive functional equation.

(c) For parameters values $\delta = 0.95$, $p = 1$ and $c = 5$, determine the optimal harvesting policy.

7.6. A firm operates in an uncertain profit environment. At the beginning of each period t, the firm observes its potential short-run variable profit π_t, which may be negative, and then decides whether to operate, making a short-run variable profit π_t, or to temporarily shut down, making a short-run variable profit of zero. Although the firm faces no fixed costs or shut-down costs, it incurs a start-up cost c if it reopens after a period of inactivity. The short-run variable profit π_t follows a stationary first-order Markov process. Specifically, short-run variable profit assumes five values p_1, p_2, p_3, p_4, and p_5 with stationary transition probabilities $P_{ij} = \Pr(\pi_{t+1} = p_j | \pi_t = p_i)$.

(a) Formulate the firm's infinite horizon profit maximization problem. Specifically, formulate Bellman's functional equation, clearly identifying the state and action variables, the state and action spaces, and the reward and probability transition functions.

(b) In the standard static model of the firm, a previously open firm will shut down if its short-run variable profit p_t is negative. Is this condition sufficient in the current model?

(c) In the standard static model of the firm, a previously closed firm will reopen if its short-run variable profit p_t exceeds the start-up cost c. Is this condition necessary in the current model?

7.7. Consider the preceding problem under the assumption that the start-up cost is $c = 0.8$, the discount factor is $\delta = 0.95$, and the short-run variable profit assumes five values $p_1 = -1.0$, $p_2 = -0.2$, $p_3 = 0.4$, $p_4 = 1.2$, and $p_5 = 2.0$ with stationary transition probabilities:

				to		
		p_1	p_2	p_3	p_4	p_4
	p_1	0.1	0.2	0.3	0.4	0.0
	p_2	0.1	0.3	0.2	0.2	0.2
from	p_3	0.1	0.5	0.2	0.1	0.1
	p_4	0.2	0.1	0.3	0.2	0.2
	p_5	0.3	0.2	0.2	0.1	0.2.

(a) Compute the optimal operation-closure policy.

(b) What is the value of the firm?

(c) In the long-run, what percentage of the time will be firm be closed?

7.8. Consider the problem of optimal harvesting of a nonrenewable resource by a competitive price-taking firm:

$$\begin{array}{ll} \max & E\sum_{t=0}^{\infty}\delta^t[p_t x_t - \alpha x_t^{\beta}] \\ \text{s.t.} & s_{t+1} = s_t - x_t \end{array}$$

where $\delta = 0.9$ is the discount factor; $\alpha = 0.2$, $\beta = 1.5$, are cost function parameters; p_t is the market price; x_t is harvest; and s_t is beginning reserves. Develop a MATLABprogram that will solve this problem numerically assuming stock and harvest levels are integers, then answer the following questions.

(a) Graph the value function for $p = 1$ and $p = 2$.

(b) Graph the optimal decision rule for $p = 1$ and $p = 2$.

(c) Assuming an initial stocks of 100 units, graph the time path of optimal harvest for periods $t = 0$ to $t = 20$, inclusive; do so for both p=1 and p=2.

(d) Under the same assumption as in (c), graph the shadow price of stocks for periods $t = 0$ to $t = 20$. Do so both in current dollars and in year 0 dollars.

7.9. Consider the preceding problem, but now assume that price takes one of two values, $p = 1$ or $p = 2$ according to the following first-order Markov probability law:

$$\begin{aligned} \Pr[p_{t+1} = 1|p_t = 1] &= 0.8 \\ \Pr[p_{t+1} = 2|p_t = 1] &= 0.2 \\ \Pr[p_{t+1} = 1|p_t = 2] &= 0.3 \\ \Pr[p_{t+1} = 2|p_t = 2] &= 0.7 \end{aligned}$$

Further assume that the manager maximizes the discounted sum of expected utility over time, where utility in year t is

$$u_t = -\exp\{-\gamma(p_t x_t - \alpha x_t^{\beta})\}$$

where $\gamma = 0.2$ is the coefficient of absolute risk aversion.

(a) Write a MATLABprogram that solves the problem.

(b) Graph the optimal decision rule for this case and for the risk neutral case on the same graph.

(c) What is the effect of risk aversion on the rate of optimal extraction in this model?

7.10. Consider the article by Burt and Allison, "Farm Management Decisions with Dynamic Programming," *Journal of Farm Economics*, 45(1963):121-37. Write a program that replicates Burt and Allison's results, then compute the optimal value function and decision rule if:

(a) the annual interest rate is 1 percent.

(b) the annual interest rate is 10 percent.

7.11. Consider Burt and Allison's farm management problem. Assume now that the government will subsidize fallow land at \$25 per acre, raising the expected return on a fallow acre from a \$2.33 loss to a \$22.67 profit. Further assume, as Burt and Allison implicitly have, that cost, price, yield, and return are determinate at each moisture level:

(a) Compute the optimal value function and decision rule.

(b) Derive the steady-state distribution of the soil moisture level under the optimal policy.

(c) Derive the steady-state distribution of return per acre under the optimal policy.

(d) Derive the steady-state mean and variance of return per acre under the optimal policy.

7.12. At the beginning of every year, a firm must decide how much to produce over the coming year in order to meet the demand for its product. The demand over any year is known at the beginning of the year, but varies annually, assuming serially independent values of 5, 6, 7, or 8 thousand units with probabilities 0.1, 0.3, 0.4, and 0.2, respectively. The firm's cost of production in year t is $10q_t + (q_t - q_{t-1})^2$ thousand dollars, where q_t is thousands of units produced in year t. The product sells for \$20 per unit and excess production can either be carried over to the following year at a cost of \$2 per unit or disposed of for free. The firm's production and storage capacities are 8 thousand and 5 thousand units per annum, respectively. The annual discount factor is 0.9. Assuming that the firm meets its annual demand exactly, and that production and storage levels must be integer multiples of one thousand units, answer the following questions:

(a) Under what conditions would the firm use all of its storage capacity?

(b) What is the value of firm and what is its optimal production if its previous year's production was 5 thousand units, its carryin is 2 thousand units, and the demand for the coming year is 7 units?

(c) What would be the production levels over the subsequent three years if the realized demands were 6, 5, and 8 units, respectively?

7.13. At dairy producer must decide whether to keep and lactate a cow or replace it with a new one. A cow yields $y_i = 8 + 2i - 0.25i^2$ tons of milk over its i^{th} lactation up to ten lactations, after which she becomes barren and must be replaced. Assume that the net cost of replacing a cow is 500 dollars, the profit contribution of milk is 150 dollars per ton, and the per-laction discount factor is $\delta = 0.9$.

(a) What lactation-replacement policy maximizes profits?

(b) What is the optimal policy if the profit contribution of milk rises to 200 dollars per ton?

(c) What is the optimal policy if the cost of replacement c_t follows a three-state Markov chain with possible values, \$400, \$500, and \$600, and transition probabilities

		c_{t+1}	
c_t	\$400	\$500	\$600
\$400	0.5	0.4	0.1
\$500	0.2	0.6	0.2
\$600	0.1	0.4	0.5